1

INTERNET MAIL PROTOCOLS

November 1982

Network Information Center
SRI International
Menlo Park, CA 94025
(NIC@SRI-NIC)

UNCLASSIFIED
SECURITY CLASSIFICATION OF THIS PAGE

REPORT DOCUMENTATION PAGE

1a. REPORT SECURITY CLASSIFICATION
U

1b. RESTRICTIVE MARKINGS

2a. SECURITY CLASSIFICATION AUTHORITY

2b. DECLASSIFICATION / DOWNGRADING SCHEDULE

3. DISTRIBUTION/AVAILABILITY OF REPORT
Approved for public release.
Distribution unlimited.

4. PERFORMING ORGANIZATION REPORT NUMBER(S)

5. MONITORING ORGANIZATION REPORT NUMBER(S)

6a. NAME OF PERFORMING ORGANIZATION
Network Information Center
SRI International

6b. OFFICE SYMBOL *(If applicable)*

7a. NAME OF MONITORING ORGANIZATION
Defense Data Network-PMO

6c. ADDRESS *(City, State, and ZIP Code)*
Menlo Park, CA 94025

7b. ADDRESS *(City, State, and ZIP Code)*
Washington, DC

8a. NAME OF FUNDING/SPONSORING ORGANIZATION

8b. OFFICE SYMBOL *(If applicable)*

9. PROCUREMENT INSTRUMENT IDENTIFICATION NUMBER
DCA200-83-C-0025

8c. ADDRESS *(City, State, and ZIP Code)*

10. SOURCE OF FUNDING NUMBERS

PROGRAM ELEMENT NO.	PROJECT NO.	TASK NO.	WORK UNIT ACCESSION NO.

11. TITLE *(Include Security Classification)*
Internet Mail Protocols

12. PERSONAL AUTHOR(S)

13a. TYPE OF REPORT

13b. TIME COVERED FROM ______ TO ______

14. DATE OF REPORT *(Year, Month, Day)*
821100

15. PAGE COUNT
125

16. SUPPLEMENTARY NOTATION

17. COSATI CODES

FIELD	GROUP	SUB-GROUP

18. SUBJECT TERMS *(Continue on reverse if necessary and identify by block number)*
Simple Mail Transfer Protocol; SMTP; Internet text message; Mail transfer; Standard; Format

19. ABSTRACT *(Continue on reverse if necessary and identify by block number)*

This document contains RFC 821(Simple Mail Transfer Protocol-SMTP) and RFC 822 (Standard for the Format of ARPA Internet Text Messages), which are the basic protocols that support all network mail. The SMTP is independent of the particular transmission subsystem, and requires only a reliable ordered data stream channel. Appendices describe the use of SMTP with various transport services. The second part of this document describes syntax requirements and options for computer messages sent by electronic mail. These RFCs update information contained in the Internet Protocol Transition Workbook. Additional Keywords: Standards; formats; Specifications; information Transfer,

20. DISTRIBUTION/AVAILABILITY OF ABSTRACT
☒ UNCLASSIFIED/UNLIMITED ☐ SAME AS RPT. ☐ DTIC USERS

21. ABSTRACT SECURITY CLASSIFICATION
Unclassified

22a. NAME OF RESPONSIBLE INDIVIDUAL

22b. TELEPHONE *(Include Area Code)*

22c. OFFICE SYMBOL

83 APR edition may be used until exhausted.
All other editions are obsolete.

SECURITY CLASSIFICATION OF THIS PAGE
UNCLASSIFIED

THE DEPARTMENT OF DEFENSE MAIL PROTOCOLS

The enclosed documents

1. RFC 821 - Simple Mail Transfer Protocol, by J. Postel, Info. Sci. Inst., Univ. S. Calif., Marina del Rey, Aug. 1982.
2. RFC 822 - Standard for the Format of ARPA Internet Text Messages, rev. by D. Crocker, Dept. Elec. Eng., Univ. Del., Newark, Aug. 13, 1982.

are updates to two documents contained in the Internet Protocol Transition Workbook dated March 1982 and distributed by the Network Information Center (NIC) on behalf of the Defense Communications Agency (DCA) and the Defense Advanced Research Projects Agency (DARPA).

This volume is being sent automatically to recipients of the Internet Protocol Transition Workbook. It may also be ordered from the NIC as a separate volume. Other DoD Internet Protocols will be sent to you as they are issued. If you no longer wish to receive these protocol updates, please contact Leda Voropaeff (415) 859-2434 or NIC@SRI-NIC and ask to have your name removed from the distribution list.

PLEASE NOTE THAT

- RFC 821 obsoletes RFC 788, RFC 780, and RFC 772
- RFC 822 obsoletes RFC 733

The mail protocols (RFC 821 and RFC 822) have been published as a separate volume because there have been numerous requests for such a volume. If you prefer, you may combine the two volumes by replacing RFC 788 and RFC 733 in the Internet Protocol Transition Workbook with RFC 821 and RFC 822 respectively.

Questions regarding the technical content of this volume should be directed to

Dr. Jon Postel
Information Sciences Institute
4676 Admiralty Way
Marina del Rey, CA 90291

(213) 822-1511
POSTEL@ISIF

Questions regarding additional copies of this volume should be directed to

Elizabeth Feinler
Network Information Center, RM EJ221
SRI International
333 Ravenswood Avenue
Menlo Park, CA 94025

(415) 859-3695
NIC@SRI-NIC

TABLE OF CONTENTS

Introduction 1

Simple Mail Transfer Protocol 5

Standard for the Format of ARPA Internet Text Messages 75

RFC 821

SIMPLE MAIL TRANSFER PROTOCOL

Jonathan B. Postel

August 1982

Information Sciences Institute
University of Southern California
4676 Admiralty Way
Marina del Rey, California 90291

(213) 822-1511

TABLE OF CONTENTS

1. INTRODUCTION 1

2. THE SMTP MODEL 2

3. THE SMTP PROCEDURE 4

3.1. Mail 4
3.2. Forwarding 7
3.3. Verifying and Expanding 8
3.4. Sending and Mailing 11
3.5. Opening and Closing 13
3.6. Relaying 14
3.7. Domains 17
3.8. Changing Roles 18

4. THE SMTP SPECIFICATIONS 19

4.1. SMTP Commands 19
4.1.1. Command Semantics 19
4.1.2. Command Syntax 27
4.2. SMTP Replies 34
4.2.1. Reply Codes by Function Group 35
4.2.2. Reply Codes in Numeric Order 36
4.3. Sequencing of Commands and Replies 37
4.4. State Diagrams 39
4.5. Details 41
4.5.1. Minimum Implementation 41
4.5.2. Transparency 41
4.5.3. Sizes 42

APPENDIX A: TCP 44
APPENDIX B: NCP 45
APPENDIX C: NITS 46
APPENDIX D: X.25 47
APPENDIX E: Theory of Reply Codes 48
APPENDIX F: Scenarios 51

GLOSSARY 64

REFERENCES 67

Network Working Group
Request for Comments: 821
Replaces: RFC 788, 780, 772

J. Postel
ISI
August 1982

SIMPLE MAIL TRANSFER PROTOCOL

1. INTRODUCTION

The objective of Simple Mail Transfer Protocol (SMTP) is to transfer mail reliably and efficiently.

SMTP is independent of the particular transmission subsystem and requires only a reliable ordered data stream channel. Appendices A, B, C, and D describe the use of SMTP with various transport services. A Glossary provides the definitions of terms as used in this document.

An important feature of SMTP is its capability to relay mail across transport service environments. A transport service provides an interprocess communication environment (IPCE). An IPCE may cover one network, several networks, or a subset of a network. It is important to realize that transport systems (or IPCEs) are not one-to-one with networks. A process can communicate directly with another process through any mutually known IPCE. Mail is an application or use of interprocess communication. Mail can be communicated between processes in different IPCEs by relaying through a process connected to two (or more) IPCEs. More specifically, mail can be relayed between hosts on different transport systems by a host on both transport systems.

2. THE SMTP MODEL

The SMTP design is based on the following model of communication: as the result of a user mail request, the sender-SMTP establishes a two-way transmission channel to a receiver-SMTP. The receiver-SMTP may be either the ultimate destination or an intermediate. SMTP commands are generated by the sender-SMTP and sent to the receiver-SMTP. SMTP replies are sent from the receiver-SMTP to the sender-SMTP in response to the commands.

Once the transmission channel is established, the SMTP-sender sends a MAIL command indicating the sender of the mail. If the SMTP-receiver can accept mail it responds with an OK reply. The SMTP-sender then sends a RCPT command identifying a recipient of the mail. If the SMTP-receiver can accept mail for that recipient it responds with an OK reply; if not, it responds with a reply rejecting that recipient (but not the whole mail transaction). The SMTP-sender and SMTP-receiver may negotiate several recipients. When the recipients have been negotiated the SMTP-sender sends the mail data, terminating with a special sequence. If the SMTP-receiver successfully processes the mail data it responds with an OK reply. The dialog is purposely lock-step, one-at-a-time.

```
-------------------------------------------------------------

               +----------+                +----------+
   +------+    |          |                |          |
   | User |<-->|          |      SMTP      |          |
   +------+    |  Sender- |Commands/Replies| Receiver-|
   +------+    |   SMTP   |<-------------->|    SMTP  |    +------+
   | File |<-->|          |    and Mail    |          |<-->| File |
   |System|    |          |                |          |    |System|
   +------+    +----------+                +----------+    +------+


                Sender-SMTP                Receiver-SMTP

                          Model for SMTP Use

                              Figure 1

-------------------------------------------------------------
```

The SMTP provides mechanisms for the transmission of mail; directly from the sending user's host to the receiving user's host when the two host are connected to the same transport service, or via one or more relay SMTP-servers when the source and destination hosts are not connected to the same transport service.

To be able to provide the relay capability the SMTP-server must be supplied with the name of the ultimate destination host as well as the destination mailbox name.

The argument to the MAIL command is a reverse-path, which specifies who the mail is from. The argument to the RCPT command is a forward-path, which specifies who the mail is to. The forward-path is a source route, while the reverse-path is a return route (which may be used to return a message to the sender when an error occurs with a relayed message).

When the same message is sent to multiple recipients the SMTP encourages the transmission of only one copy of the data for all the recipients at the same destination host.

The mail commands and replies have a rigid syntax. Replies also have a numeric code. In the following, examples appear which use actual commands and replies. The complete lists of commands and replies appears in Section 4 on specifications.

Commands and replies are not case sensitive. That is, a command or reply word may be upper case, lower case, or any mixture of upper and lower case. Note that this is not true of mailbox user names. For some hosts the user name is case sensitive, and SMTP implementations must take case to preserve the case of user names as they appear in mailbox arguments. Host names are not case sensitive.

Commands and replies are composed of characters from the ASCII character set [1]. When the transport service provides an 8-bit byte (octet) transmission channel, each 7-bit character is transmitted right justified in an octet with the high order bit cleared to zero.

When specifying the general form of a command or reply, an argument (or special symbol) will be denoted by a meta-linguistic variable (or constant), for example, "<string>" or "<reverse-path>". Here the angle brackets indicate these are meta-linguistic variables. However, some arguments use the angle brackets literally. For example, an actual reverse-path is enclosed in angle brackets, i.e., "<John.Smith@USC-ISI.ARPA>" is an instance of <reverse-path> (the angle brackets are actually transmitted in the command or reply).

3. THE SMTP PROCEDURES

This section presents the procedures used in SMTP in several parts. First comes the basic mail procedure defined as a mail transaction. Following this are descriptions of forwarding mail, verifying mailbox names and expanding mailing lists, sending to terminals instead of or in combination with mailboxes, and the opening and closing exchanges. At the end of this section are comments on relaying, a note on mail domains, and a discussion of changing roles. Throughout this section are examples of partial command and reply sequences, several complete scenarios are presented in Appendix F.

3.1. MAIL

There are three steps to SMTP mail transactions. The transaction is started with a MAIL command which gives the sender identification. A series of one or more RCPT commands follows giving the receiver information. Then a DATA command gives the mail data. And finally, the end of mail data indicator confirms the transaction.

The first step in the procedure is the MAIL command. The <reverse-path> contains the source mailbox.

```
MAIL <SP> FROM:<reverse-path> <CRLF>
```

This command tells the SMTP-receiver that a new mail transaction is starting and to reset all its state tables and buffers, including any recipients or mail data. It gives the reverse-path which can be used to report errors. If accepted, the receiver-SMTP returns a 250 OK reply.

The <reverse-path> can contain more than just a mailbox. The <reverse-path> is a reverse source routing list of hosts and source mailbox. The first host in the <reverse-path> should be the host sending this command.

The second step in the procedure is the RCPT command.

```
RCPT <SP> TO:<forward-path> <CRLF>
```

This command gives a forward-path identifying one recipient. If accepted, the receiver-SMTP returns a 250 OK reply, and stores the forward-path. If the recipient is unknown the receiver-SMTP returns a 550 Failure reply. This second step of the procedure can be repeated any number of times.

The <forward-path> can contain more than just a mailbox. The <forward-path> is a source routing list of hosts and the destination mailbox. The first host in the <forward-path> should be the host receiving this command.

The third step in the procedure is the DATA command.

```
DATA <CRLF>
```

If accepted, the receiver-SMTP returns a 354 Intermediate reply and considers all succeeding lines to be the message text. When the end of text is received and stored the SMTP-receiver sends a 250 OK reply.

Since the mail data is sent on the transmission channel the end of the mail data must be indicated so that the command and reply dialog can be resumed. SMTP indicates the end of the mail data by sending a line containing only a period. A transparency procedure is used to prevent this from interfering with the user's text (see Section 4.5.2).

> Please note that the mail data includes the memo header items such as Date, Subject, To, Cc, From [2].

The end of mail data indicator also confirms the mail transaction and tells the receiver-SMTP to now process the stored recipients and mail data. If accepted, the receiver-SMTP returns a 250 OK reply. The DATA command should fail only if the mail transaction was incomplete (for example, no recipients), or if resources are not available.

The above procedure is an example of a mail transaction. These commands must be used only in the order discussed above. Example 1 (below) illustrates the use of these commands in a mail transaction.

Example of the SMTP Procedure

This SMTP example shows mail sent by Smith at host Alpha.ARPA, to Jones, Green, and Brown at host Beta.ARPA. Here we assume that host Alpha contacts host Beta directly.

```
S: MAIL FROM:<Smith@Alpha.ARPA>
R: 250 OK

S: RCPT TO:<Jones@Beta.ARPA>
R: 250 OK

S: RCPT TO:<Green@Beta.ARPA>
R: 550 No such user here

S: RCPT TO:<Brown@Beta.ARPA>
R: 250 OK

S: DATA
R: 354 Start mail input; end with <CRLF>.<CRLF>
S: Blah blah blah...
S: ...etc. etc. etc.
S: <CRLF>.<CRLF>
R: 250 OK
```

The mail has now been accepted for Jones and Brown. Green did not have a mailbox at host Beta.

Example 1

3.2. FORWARDING

There are some cases where the destination information in the <forward-path> is incorrect, but the receiver-SMTP knows the correct destination. In such cases, one of the following replies should be used to allow the sender to contact the correct destination.

251 User not local; will forward to <forward-path>

This reply indicates that the receiver-SMTP knows the user's mailbox is on another host and indicates the correct forward-path to use in the future. Note that either the host or user or both may be different. The receiver takes responsibility for delivering the message.

551 User not local; please try <forward-path>

This reply indicates that the receiver-SMTP knows the user's mailbox is on another host and indicates the correct forward-path to use. Note that either the host or user or both may be different. The receiver refuses to accept mail for this user, and the sender must either redirect the mail according to the information provided or return an error response to the originating user.

Example 2 illustrates the use of these responses.

Example of Forwarding

Either

```
S: RCPT TO:<Postel@USC-ISI.ARPA>
R: 251 User not local; will forward to <Postel@USC-ISIF.ARPA>
```

Or

```
S: RCPT TO:<Paul@USC-ISIB.ARPA>
R: 551 User not local; please try <Mockapetris@USC-ISIF.ARPA>
```

Example 2

3.3. VERIFYING AND EXPANDING

SMTP provides as additional features, commands to verify a user name or expand a mailing list. This is done with the VRFY and EXPN commands, which have character string arguments. For the VRFY command, the string is a user name, and the response may include the full name of the user and must include the mailbox of the user. For the EXPN command, the string identifies a mailing list, and the multiline response may include the full name of the users and must give the mailboxes on the mailing list.

"User name" is a fuzzy term and used purposely. If a host implements the VRFY or EXPN commands then at least local mailboxes must be recognized as "user names". If a host chooses to recognize other strings as "user names" that is allowed.

In some hosts the distinction between a mailing list and an alias for a single mailbox is a bit fuzzy, since a common data structure may hold both types of entries, and it is possible to have mailing lists of one mailbox. If a request is made to verify a mailing list a positive rosponse can be given if on receipt of a message so addressed it will be delivered to everyone on the list, otherwise an error should be reported (e.g., "550 That is a mailing list, not a user"). If a request is made to expand a user name a positive response can be formed by returning a list containing one name, or an error can be reported (e.g., "550 That is a user name, not a mailing list").

In the case of a multiline reply (normal for EXPN) exactly one mailbox is to be specified on each line of the reply. In the case of an ambiguous request, for example, "VRFY Smith", where there are two Smith's the response must be "553 User ambiguous".

The case of verifying a user name is straightforward as shown in example 3.

```
-------------------------------------------------------------

              Example of Verifying a User Name

   Either

      S: VRFY Smith
      R: 250 Fred Smith <Smith@USC-ISIF.ARPA>

   Or

      S: VRFY Smith
      R: 251 User not local; will forward to <Smith@USC-ISIQ.ARPA>

   Or

      S: VRFY Jones
      R: 550 String does not match anything.

   Or

      S: VRFY Jones
      R: 551 User not local; please try <Jones@USC-ISIQ.ARPA>

   Or

      S: VRFY Gourzenkyinplatz
      R: 553 User ambiguous.

                          Example 3

-------------------------------------------------------------
```

The case of expanding a mailbox list requires a multiline reply as shown in example 4.

Example of Expanding a Mailing List

```
Either

   S: EXPN Example-People
   R: 250-Jon Postel <Postel@USC-ISIF.ARPA>
   R: 250-Fred Fonebcne <Fonebone@USC-ISIQ.ARPA>
   R: 250-Sam Q. Smith <SQSmith@USC-ISIC.ARPA>
   R: 250-Quincy Smith <@USC-ISIF.ARPA:Q-Smith@ISI-VAXA.ARPA>
   R: 250-<joe@foo-unix.ARPA>
   R: 250 <xyz@bar-unix.ARPA>

Or

   S: EXPN Executive-Washroom-List
   R: 550 Access Denied to You.
```

Example 4

The character string arguments of the VRFY and EXPN commands cannot be further restricted due to the variety of implementations of the user name and mailbox list concepts. On some systems it may be appropriate for the argument of the EXPN command to be a file name for a file containing a mailing list, but again there is a variety of file naming conventions in the Internet.

The VRFY and EXPN commands are not included in the minimum implementation (Section 4.5.1), and are not required to work across relays when they are implemented.

3.4. SENDING AND MAILING

The main purpose of SMTP is to deliver messages to user's mailboxes. A very similar service provided by some hosts is to deliver messages to user's terminals (provided the user is active on the host). The delivery to the user's mailbox is called "mailing", the delivery to the user's terminal is called "sending". Because in many hosts the implementation of sending is nearly identical to the implementation of mailing these two functions are combined in SMTP. However the sending commands are not included in the required minimum implementation (Section 4.5.1). Users should have the ability to control the writing of messages on their terminals. Most hosts permit the users to accept or refuse such messages.

The following three command are defined to support the sending options. These are used in the mail transaction instead of the MAIL command and inform the receiver-SMTP of the special semantics of this transaction:

SEND <SP> FROM:<reverse-path> <CRLF>

The SEND command requires that the mail data be delivered to the user's terminal. If the user is not active (or not accepting terminal messages) on the host a 450 reply may returned to a RCPT command. The mail transaction is successful if the message is delivered the terminal.

SOML <SP> FROM:<reverse-path> <CRLF>

The Send Or MaiL command requires that the mail data be delivered to the user's terminal if the user is active (and accepting terminal messages) on the host. If the user is not active (or not accepting terminal messages) then the mail data is entered into the user's mailbox. The mail transaction is successful if the message is delivered either to the terminal or the mailbox.

SAML <SP> FROM:<reverse-path> <CRLF>

The Send And MaiL command requires that the mail data be delivered to the user's terminal if the user is active (and accepting terminal messages) on the host. In any case the mail data is entered into the user's mailbox. The mail transaction is successful if the message is delivered the mailbox.

The same reply codes that are used for the MAIL commands are used for these commands.

3.5. OPENING AND CLOSING

At the time the transmission channel is opened there is an exchange to ensure that the hosts are communicating with the hosts they think they are.

The following two commands are used in transmission channel opening and closing:

```
HELO <SP> <domain> <CRLF>

QUIT <CRLF>
```

In the HELO command the host sending the command identifies itself; the command may be interpreted as saying "Hello, I am <domain>".

Example of Connection Opening

```
R: 220 BBN-UNIX.ARPA Simple Mail Transfer Service Ready
S: HELO USC-ISIF.ARPA
R: 250 BBN-UNIX.ARPA
```

Example 5

Example of Connection Closing

```
S: QUIT
R: 221 BBN-UNIX.ARPA Service clcsing transmission channel
```

Example 6

3.6. RELAYING

The forward-path may be a source route of the form "@ONE,@TWO:JOE@THREE", where ONE, TWO, and THREE are hosts. This form is used to emphasize the distinction between an address and a route. The mailbox is an absolute address, and the route is information about how to get there. The two concepts should not be confused.

Conceptually the elements of the forward-path are moved to the reverse-path as the message is relayed from one server-SMTP to another. The reverse-path is a reverse source route, (i.e., a source route from the current location of the message to the originator of the message). When a server-SMTP deletes its identifier from the forward-path and inserts it into the reverse-path, it must use the name it is known by in the environment it is sending into, not the environment the mail came from, in case the server-SMTP is known by different names in different environments.

If when the message arrives at an SMTP the first element of the forward-path is not the identifier of that SMTP the element is not deleted from the forward-path and is used to determine the next SMTP to send the message to. In any case, the SMTP adds its own identifier to the reverse-path.

Using source routing the receiver-SMTP receives mail to be relayed to another server-SMTP The receiver-SMTP may accept or reject the task of relaying the mail in the same way it accepts or rejects mail for a local user. The receiver-SMTP transforms the command arguments by moving its own identifier from the forward-path to the beginning of the reverse-path. The receiver-SMTP then becomes a sender-SMTP, establishes a transmission channel to the next SMTP in the forward-path, and sends it the mail.

The first host in the reverse-path should be the host sending the SMTP commands, and the first host in the forward-path should be the host receiving the SMTP commands.

Notice that the forward-path and reverse-path appear in the SMTP commands and replies, but not necessarily in the message. That is, there is no need for these paths and especially this syntax to appear in the "To:" , "From:", "CC:", etc. fields of the message header.

If a server-SMTP has accepted the task of relaying the mail and later finds that the forward-path is incorrect or that the mail cannot be delivered for whatever reason, then it must construct an "undeliverable mail" notification message and send it to the originator of the undeliverable mail (as indicated by the reverse-path).

This notification message must be from the server-SMTP at this host. Of course, server-SMTPs should not send notification messages about problems with notification messages. One way to prevent loops in error reporting is to specify a null reverse-path in the MAIL command of a notification message. When such a message is relayed it is permissible to leave the reverse-path null. A MAIL command with a null reverse-path appears as follows:

```
MAIL FROM:<>
```

An undeliverable mail notification message is shown in example 7. This notification is in response to a message originated by JOE at HOSTW and sent via HOSTX to HOSTY with instructions to relay it on to HOSTZ. What we see in the example is the transaction between HOSTY and HOSTX, which is the first step in the return of the notification message.

Example Undeliverable Mail Notification Message

```
S: MAIL FROM:<>
R: 250 ok
S: RCPT TO:<@HOSTX.ARPA:JOE@HOSTW.ARPA>
R: 250 ok
S: DATA
R: 354 send the mail data, end with .
S: Date: 23 Oct 81 11:22:33
S: From: SMTP@HOSTY.ARPA
S: To: JOE@HOSTW.ARPA
S: Subject: Mail System Problem
S:
S:   Sorry JOE, your message to SAM@HOSTZ.ARPA lost.
S:   HOSTZ.ARPA said this:
S:    "550 No Such User"
S: .
R: 250 ok
```

Example 7

3.7. DOMAINS

Domains are a recently introduced concept in the ARPA Internet mail system. The use of domains changes the address space from a flat global space of simple character string host names to a hierarchically structured rooted tree of global addresses. The host name is replaced by a domain and host designator which is a sequence of domain element strings separated by periods with the understanding that the domain elements are ordered from the most specific to the most general.

For example, "USC-ISIF.ARPA", "Fred.Cambridge.UK", and "PC7.LCS.MIT.ARPA" might be host-and-domain identifiers.

Whenever domain names are used in SMTP only the official names are used, the use of nicknames or aliases is not allowed.

3.8. CHANGING ROLES

The TURN command may be used to reverse the roles of the two programs communicating over the transmission channel.

If program-A is currently the sender-SMTP and it sends the TURN command and receives an ok reply (250) then program-A becomes the receiver-SMTP.

If program-B is currently the receiver-SMTP and it receives the TURN command and sends an ok reply (250) then program-B becomes the sender-SMTP.

To refuse to change roles the receiver sends the 502 reply.

Please note that this command is optional. It would not normally be used in situations where the transmission channel is TCP. However, when the cost of establishing the transmission channel is high, this command may be quite useful. For example, this command may be useful in supporting be mail exchange using the public switched telephone system as a transmission channel, especially if some hosts poll other hosts for mail exchanges.

4. THE SMTP SPECIFICATIONS

4.1. SMTP COMMANDS

4.1.1. COMMAND SEMANTICS

The SMTP commands define the mail transfer or the mail system function requested by the user. SMTP commands are character strings terminated by <CRLF>. The command codes themselves are alphabetic characters terminated by <SP> if paramoters follow and <CRLF> otherwise. The syntax of mailboxes must conform to receiver site conventions. The SMTP commands are discussed below. The SMTP replies are discussed in the Section 4.2.

A mail transaction involves several data objects which are communicated as arguments to different commands. The reverse-path is the argument of the MAIL command, the forward-path is the argument of the RCPT command, and the mail data is the argument of the DATA command. These arguments or data objects must be transmitted and held pending the confirmation communicated by the end of mail data indication which finalizes the transaction. The model for this is that distinct buffers are provided to hold the types of data objects, that is, there is a reverse-path buffer, a forward-path buffer, and a mail data buffer. Specific commands cause information to be appended to a spacific buffer, or cause one or more buffers to be cleared.

HELLO (HELO)

This command is used to identify the sender-SMTP to the receiver-SMTP. The argument field contains the host name of the sender-SMTP.

The receiver-SMTP identifies itself to the sender-SMTP in the connection greeting reply, and in the response to this command.

This command and an OK reply to it confirm that both the sender-SMTP and the receiver-SMTP are in the initial state, that is, there is no transaction in progress and all state tables and buffers are cleared.

MAIL (MAIL)

This command is used to initiate a mail transaction in which the mail data is delivered to one or more mailboxes. The argument field contains a reverse-path.

The reverse-path consists of an optional list of hosts and the sender mailbox. When the list of hosts is present, it is a "reverse" source route and indicates that the mail was relayed through each host on the list (the first host in the list was the most recent relay). This list is used as a source route to return non-delivery notices to the sender. As each relay host adds itself to the beginning of the list, it must use its name as known in the IPCE to which it is relaying the mail rather than the IPCE from which the mail came (if they are different). In some types of error reporting messages (for example, undeliverable mail notifications) the reverse-path may be null (see Example 7).

This command clears the reverse-path buffer, the forward-path buffer, and the mail data buffer; and inserts the reverse-path information from this command into the reverse-path buffer.

RECIPIENT (RCPT)

This command is used to identify an individual recipient of the mail data; multiple recipients are specified by multiple use of this command.

The forward-path consists of an optional list of hosts and a required destination mailbox. When the list of hosts is present, it is a source route and indicates that the mail must be relayed to the next host on the list. If the receiver-SMTP does not implement the relay function it may user the same reply it would for an unknown local user (550).

When mail is relayed, the relay host must remove itself from the beginning forward-path and put itself at the beginning of the reverse-path. When mail reaches its ultimate destination (the forward-path contains only a destination mailbox), the receiver-SMTP inserts it into the destination mailbox in accordance with its host mail conventions.

For example, mail received at relay host A with arguments

```
FROM:<USERX@HOSTY.ARPA>
TO:<@HOSTA.ARPA,@HOSTB.ARPA:USERC@HOSTD.ARPA>
```

will be relayed on to host B with arguments

```
FROM:<@HOSTA.ARPA:USERX@HOSTY.ARPA>
TO:<@HOSTB.ARPA:USERC@HOSTD.ARPA>.
```

This command causes its forward-path argument to be appended to the forward-path buffer.

DATA (DATA)

The receiver treats the lines following the command as mail data from the sender. This command causes the mail data from this command to be appended to the mail data buffer. The mail data may contain any of the 128 ASCII character codes.

The mail data is terminated by a line containing only a period, that is the character sequence "<CRLF>.<CRLF>" (see Section 4.5.2 on Transparency). This is the end of mail data indication.

The end of mail data indication requires that the receiver must now process the stored mail transaction information. This processing consumes the information in the reverse-path buffer, the forward-path buffer, and the mail data buffer, and on the completion of this command these buffers are cleared. If the processing is successful the receiver must send an OK reply. If the processing fails completely the receiver must send a failure reply.

When the receiver-SMTP accepts a message either for relaying or for final delivery it inserts at the beginning of the mail data a time stamp line. The time stamp line indicates the identity of the host that sent the message, and the identity of the host that received the message (and is inserting this time stamp), and the date and time the message was received. Relayed messages will have multiple time stamp lines.

When the receiver-SMTP makes the "final delivery" of a message it inserts at the beginning of the mail data a return path line. The return path line preserves the information in the <reverse-path> from the MAIL command. Here, final delivery means the message leaves the SMTP world. Normally, this would mean it has been delivered to the destination user, but in some cases it may be further processed and transmitted by another mail system.

It is possible for the mailbox in the return path be different from the actual sender's mailbox, for example, if error responses are to be delivered a special error handling mailbox rather than the message senders.

The preceding two paragraphs imply that the final mail data will begin with a return path line, followed by one or more time stamp lines. These lines will be followed by the mail data header and body [2]. See Example 8.

Special mention is needed of the response and further action required when the processing following the end of mail data indication is partially successful. This could arise if after accepting several recipients and the mail data, the receiver-SMTP finds that the mail data can be successfully delivered to some of the recipients, but it cannot be to others (for example, due to mailbox space allocation problems). In such a situation, the response to the DATA command must be an OK reply. But, the receiver-SMTP must compose and send an "undeliverable mail" notification message to the originator of the message. Either a single notification which lists all of the recipients that failed to get the message, or separate notification messages must be sent for each failed recipient (see Example 7). All undeliverable mail notification messages are sent using the MAIL command (even if they result from processing a SEND, SOML, or SAML command).

Example of Return Path and Received Time Stamps

```
Return-Path: <@GHI.ARPA,@DEF.ARPA,@ABC.ARPA:JOE@ABC.ARPA>
Received: from GHI.ARPA by JKL.ARPA ; 27 Oct 81 15:27:39 PST
Received: from DEF.ARPA by GHI.ARPA ; 27 Oct 81 15:15:13 PST
Received: from ABC.ARPA by DEF.ARPA ; 27 Oct 81 15:01:59 PST
Date: 27 Oct 81 15:01:01 PST
From: JOE@ABC.ARPA
Subject: Improved Mailing System Installed
To: SAM@JKL.ARPA

This is to inform you that ...
```

Example 8

SEND (SEND)

This command is used to initiate a mail transaction in which the mail data is delivered to one or more terminals. The argument field contains a reverse-path. This command is successful if the message is delivered to a terminal.

The reverse-path consists of an optional list of hosts and the sender mailbox. When the list of hosts is present, it is a "reverse" source route and indicates that the mail was relayed through each host on the list (the first host in the list was the most recent relay). This list is used as a source route to return non-delivery notices to the sender. As each relay host adds itself to the beginning of the list, it must use its name as known in the IPCE to which it is relaying the mail rather than the IPCE from which the mail came (if they are different).

This command clears the reverse-path buffer, the forward-path buffer, and the mail data buffer; and inserts the reverse-path information from this command into the reverse-path buffer.

SEND OR MAIL (SOML)

This command is used to initiate a mail transaction in which the mail data is delivered to one or more terminals or mailboxes. For each recipient the mail data is delivered to the recipient's terminal if the recipient is active on the host (and accepting terminal messages), otherwise to the recipient's mailbox. The argument field contains a reverse-path. This command is successful if the message is delivered to a terminal or the mailbox.

The reverse-path consists of an optional list of hosts and the sender mailbox. When the list of hosts is present, it is a "reverse" source route and indicates that the mail was relayed through each host on the list (the first host in the list was the most recent relay). This list is used as a source route to return non-delivery notices to the sender. As each relay host adds itself to the beginning of the list, it must use its name as known in the IPCE to which it is relaying the mail rather than the IPCE from which the mail came (if they are different).

This command clears the reverse-path buffer, the forward-path buffer, and the mail data buffer; and inserts the reverse-path information from this command into the reverse-path buffer.

SEND AND MAIL (SAML)

This command is used to initiate a mail transaction in which the mail data is delivered to one or more terminals and mailboxes. For each recipient the mail data is delivered to the recipient's terminal if the recipient is active on the host (and accepting terminal messages), and for all recipients to the recipient's mailbox. The argument field contains a reverse-path. This command is successful if the message is delivered to the mailbox.

The reverse-path consists of an optional list of hosts and the sender mailbox. When the list of hosts is present, it is a "reverse" source route and indicates that the mail was relayed through each host on the list (the first host in the list was the most recent relay). This list is used as a source route to return non-delivery notices to the sender. As each relay host adds itself to the beginning of the list, it must use its name as known in the IPCE to which it is relaying the mail rather than the IPCE from which the mail came (if they are different).

This command clears the reverse-path buffer, the forward-path buffer, and the mail data buffer; and inserts the reverse-path information from this command into the reverse-path buffer.

RESET (RSET)

This command specifies that the current mail transaction is to be aborted. Any stored sender, recipients, and mail data must be discarded, and all buffers and state tables cleared. The receiver must send an OK reply.

VERIFY (VRFY)

This command asks the receiver to confirm that the argument identifies a user. If it is a user name, the full name of the user (if known) and the fully specified mailbox are returned.

This command has no effect on any of the reverse-path buffer, the forward-path buffer, or the mail data buffer.

EXPAND (EXPN)

This command asks the receiver to confirm that the argument identifies a mailing list, and if so, to return the membership of that list. The full name of the users (if known) and the fully specified mailboxes are returned in a multiline reply.

This command has no effect on any of the reverse-path buffer, the forward-path buffer, or the mail data buffer.

HELP (HELP)

This command causes the receiver to send helpful information to the sender of the HELP command. The command may take an argument (e.g., any command name) and return more specific information as a response.

This command has no effect on any of the reverse-path buffer, the forward-path buffer, or the mail data buffer.

NOOP (NOOP)

This command does not affect any parameters or previously entered commands. It specifies no action other than that the receiver send an OK reply.

This command has no effect on any of the reverse-path buffer, the forward-path buffer, or the mail data buffer.

QUIT (QUIT)

This command specifies that the receiver must send an OK reply, and then close the transmission channel.

The receiver should not close the transmission channel until it receives and replies to a QUIT command (even if there was an error). The sender should not close the transmission channel until it send a QUIT command and receives the reply (even if there was an error response to a previous command). If the connection is closed prematurely the receiver should act as if a RSET command had been received (canceling any pending transaction, but not undoing any previously completed transaction), the sender should act as if the command or transaction in progress had received a temporary error (4xx).

TURN (TURN)

This command specifies that the receiver must either (1) send an OK reply and then take on the role of the sender-SMTP, or (2) send a refusal reply and retain the role of the receiver-SMTP.

If program-A is currently the sender-SMTP and it sends the TURN command and receives an OK reply (250) then program-A becomes the receiver-SMTP. Program-A is then in the initial state as if the transmission channel just opened, and it then sends the 220 service ready greeting.

If program-B is currently the receiver-SMTP and it receives the TURN command and sends an OK reply (250) then program-B becomes the sender-SMTP. Program-B is then in the initial state as if the transmission channel just opened, and it then expects to receive the 220 service ready greeting.

To refuse to change roles the receiver sends the 502 reply.

There are restrictions on the order in which these command may be used.

The first command in a session must be the HELO command. The HELO command may be used later in a session as well. If the HELO command argument is not acceptable a 501 failure reply must be returned and the receiver-SMTP must stay in the same state.

The NOOP, HELP, EXPN, and VRFY commands can be used at any time during a session.

The MAIL, SEND, SOML, or SAML commands begin a mail transaction. Once started a mail transaction consists of one of the transaction beginning commands, one or more RCPT commands, and a DATA command, in that order. A mail transaction may be aborted by the RSET command. There may be zero or more transactions in a session.

If the transaction beginning command argument is not acceptable a 501 failure reply must be returned and the receiver-SMTP must stay in the same state. If the commands in a transaction are out of order a 503 failure reply must be returned and the receiver-SMTP must stay in the same state.

The last command in a session must be the QUIT command. The QUIT command can not be used at any other time in a session.

4.1.2. COMMAND SYNTAX

The commands consist of a command code followed by an argument field. Command codes are four alphabetic characters. Upper and lower case alphabetic characters are to be treated identically. Thus, any of the following may represent the mail command:

```
MAIL    Mail    mail    MaIl    mAIl
```

This also applies to any symbols representing parameter values, such as "TO" or "to" for the forward-path. Command codes and the argument fields are separated by one or more spaces. However, within the reverse-path and forward-path arguments case is important. In particular, in some hosts the user "smith" is different from the user "Smith".

The argument field consists of a variable length character string ending with the character sequence <CRLF>. The receiver is to take no action until this sequence is received.

Square brackets denote an optional argument field. If the option is not taken, the appropriate default is implied.

The following are the SMTP commands:

```
HELO <SP> <domain> <CRLF>

MAIL <SP> FROM:<reverse-path> <CRLF>

RCPT <SP> TO:<forward-path> <CRLF>

DATA <CRLF>

RSET <CRLF>

SEND <SP> FROM:<reverse-path> <CRLF>

SOML <SP> FROM:<reverse-path> <CRLF>

SAML <SP> FROM:<reverse-path> <CRLF>

VRFY <SP> <string> <CRLF>

EXPN <SP> <string> <CRLF>

HELP [<SP> <string>] <CRLF>

NOOP <CRLF>

QUIT <CRLF>

TURN <CRLF>
```

The syntax of the above argument fields (using BNF notation where applicable) is given below. The "..." notation indicates that a field may be repeated one or more times.

```
<reverse-path> ::= <path>

<forward-path> ::= <path>

<path> ::= "<" [ <a-d-l> ":" ] <mailbox> ">"

<a-d-l> ::= <at-domain> | <at-domain> "," <a-d-l>

<at-domain> ::= "@" <domain>

<domain> ::=  <element> | <element> "." <domain>

<element> ::= <name> | "#" <number> | "[" <dotnum> "]"

<mailbox> ::= <local-part> "@" <domain>

<local-part> ::= <dot-string> | <quoted-string>

<name> ::= <a> <ldh-str> <let-dig>

<ldh-str> ::= <let-dig-hyp> | <let-dig-hyp> <ldh-str>

<let-dig> ::= <a> | <d>

<let-dig-hyp> ::= <a> | <d> | "-"

<dot-string> ::= <string> | <string> "." <dot-string>

<string> ::= <char> | <char> <string>

<quoted-string> ::=  """ <qtext> """

<qtext> ::=  "\" <x> | "\" <x> <qtext> | <q> | <q> <qtext>

<char> ::= <c> | "\" <x>

<dotnum> ::= <snum> "." <snum> "." <snum> "." <snum>

<number> ::= <d> | <d> <number>

<CRLF> ::= <CR> <LF>
```

```
<CR> ::= the carriage return character (ASCII code 13)

<LF> ::= the line feed character (ASCII code 10)

<SP> ::= the space character (ASCII code 32)

<snum> ::= one, two, or three digits representing a decimal
          integer value in the range 0 through 255

<a> ::= any one of the 52 alphabetic characters A through Z
          in upper case and a through z in lower case

<c> ::= any one of the 128 ASCII characters, but not any
          <special> or <SP>

<d> ::= any one of the ten digits 0 through 9

<q> ::= any one of the 128 ASCII characters except <CR>,
          <LF>, quote ("), or backslash (\)

<x> ::= any one of the 128 ASCII characters (no exceptions)

<special> ::= "<" | ">" | "(" | ")" | "[" | "]" | "\" | "."
          | "," | ";" | ":" | "@"  """ | the control
          characters (ASCII codes 0 through 31 inclusive and
          127)
```

Note that the backslash, "\", is a quote character, which is used to indicate that the next character is to be used literally (instead of its normal interpretation). For example, "Joe\,Smith" could be used to indicate a single nine character user field with comma being the fourth character of the field.

Hosts are generally known by names which are translated to addresses in each host. Note that the name elements of domains are the official names -- no use of nicknames or aliases is allowed.

Sometimes a host is not known to the translation function and communication is blocked. To bypass this barrier two numeric forms are also allowed for host "names". One form is a decimal integer prefixed by a pound sign, "#", which indicates the number is the address of the host. Another form is four small decimal integers separated by dots and enclosed by brackets, e.g., "[123.255.37.2]", which indicates a 32-bit ARPA Internet Address in four 8-bit fields.

The time stamp line and the return path line are formally defined as follows:

```
<return-path-line> ::= "Return-Path:" <SP><reverse-path><CRLF>

<time-stamp-line> ::= "Received:" <SP> <stamp> <CRLF>

   <stamp> ::= <from-domain> <by-domain> <opt-info> ";"
             <daytime>

   <from-domain> ::= "FROM" <SP> <domain> <SP>

   <by-domain> ::= "BY" <SP> <domain> <SP>

   <opt-info> ::= [<via>] [<with>] [<id>] [<for>]

   <via> ::= "VIA" <SP> <link> <SP>

   <with> ::= "WITH" <SP> <protocol> <SP>

   <id> ::= "ID" <SP> <string> <SP>

   <for> ::= "FOR" <SP> <path> <SP>

   <link> ::= The standard names for links are registered with
             the Network Information Center.

   <protocol> ::= The standard names for protocols are
             registered with the Network Information Center.

   <daytime> ::= <SP> <date> <SP> <time>

   <date> ::= <dd> <SP> <mon> <SP> <yy>

   <time> ::= <hh> ":" <mm> ":" <ss> <SP> <zone>

   <dd> ::= the one or two decimal integer day of the month in
             the range 1 to 31.

   <mon> ::= "JAN" | "FEB" | "MAR" | "APR" | "MAY" | "JUN" |
             "JUL" | "AUG" | "SEP" | "OCT" | "NOV" | "DEC"

   <yy> ::= the two decimal integer year of the century in the
             range 00 to 99.
```

```
<hh> ::= the two decimal integer hour of the day in the
         range 00 to 24.

<mm> ::= the two decimal integer minute of the hour in the
         range 00 to 59.

<ss> ::= the two decimal integer second of the minute in the
         range 00 to 59.

<zone> ::= "UT" for Universal Time (the default) or other
           time zone designator (as in [2]).
```

Return Path Example

```
Return-Path: <@CHARLIE.ARPA,@BAKER.ARPA:JOE@ABLE.ARPA>
```

Example 9

Time Stamp Line Example

```
Received: FROM ABC.ARPA BY XYZ.ARPA ; 22 OCT 81 09:23:59 PDT

   Received: from ABC.ARPA by XYZ.ARPA via TELENET with X25
             id M12345 for Smith@PDQ.ARPA ; 22 OCT 81 09:23:59 PDT
```

Example 10

4.2. SMTP REPLIES

Replies to SMTP commands are devised to ensure the synchronization of requests and actions in the process of mail transfer, and to guarantee that the sender-SMTP always knows the state of the receiver-SMTP. Every command must generate exactly one reply.

The details of the command-reply sequence are made explicit in Section 5.3 on Sequencing and Section 5.4 State Diagrams.

An SMTP reply consists of a three digit number (transmitted as three alphanumeric characters) followed by some text. The number is intended for use by automata to determine what state to enter next; the text is meant for the human user. It is intended that the three digits contain enough encoded information that the sender-SMTP need not examine the text and may either discard it or pass it on to the user, as appropriate. In particular, the text may be receiver-dependent and context dependent, so there are likely to be varying texts for each reply code. A discussion of the theory of reply codes is given in Appendix E. Formally, a reply is defined to be the sequence: a three-digit code, <SP>, one line of text, and <CRLF>, or a multiline reply (as defined in Appendix E). Only the EXPN and HELP commands are expected to result in multiline replies in normal circumstances, however multiline replies are allowed for any command.

4.2.1. REPLY CODES BY FUNCTION GROUPS

```
500 Syntax error, command unrecognized
   [This may include errors such as command line too long]
501 Syntax error in parameters or arguments
502 Command not implemented
503 Bad sequence of commands
504 Command parameter not implemented

211 System status, or system help reply
214 Help message
   [Information on how to use the receiver or the meaning of a
   particular non-standard command; this reply is useful only
   to the human user]

220 <domain> Service ready
221 <domain> Service closing transmission channel
421 <domain> Service not available,
    closing transmission channel
   [This may be a reply to any command if the service knows it
   must shut down]

250 Requested mail action okay, completed
251 User not local; will forward to <forward-path>
450 Requested mail action not taken: mailbox unavailable
    [E.g., mailbox busy]
550 Requested action not taken: mailbox unavailable
    [E.g., mailbox not found, no access]
451 Requested action aborted: error in processing
551 User not local; please try <forward-path>
452 Requested action not taken: insufficient system storage
552 Requested mail action aborted: exceeded storage allocation
553 Requested action not taken: mailbox name not allowed
    [E.g., mailbox syntax incorrect]
354 Start mail input; end with <CRLF>.<CRLF>
554 Transaction failed
```

4.2.2. NUMERIC ORDER LIST OF REPLY CODES

```
211 System status, or system help reply
214 Help message
   [Information on how to use the receiver or the meaning of a
   particular non-standard command; this reply is useful only
   to the human user]
220 <domain> Service ready
221 <domain> Service closing transmission channel
250 Requested mail action okay, completed
251 User not local; will forward to <forward-path>

354 Start mail input; end with <CRLF>.<CRLF>

421 <domain> Service not available,
    closing transmission channel
   [This may be a reply to any command if the service knows it
   must shut down]
450 Requested mail action not taken: mailbox unavailable
   [E.g., mailbox busy]
451 Requested action aborted: local error in processing
452 Requested action not taken: insufficient system storage

500 Syntax error, command unrecognized
   [This may include errors such as command line too long]
501 Syntax error in parameters or arguments
502 Command not implemented
503 Bad sequence of commands
504 Command parameter not implemented
550 Requested action not taken: mailbox unavailable
   [E.g., mailbox not found, no access]
551 User not local; please try <forward-path>
552 Requested mail action aborted: exceeded storage allocation
553 Requested action not taken: mailbox name not allowed
   [E.g., mailbox syntax incorrect]
554 Transaction failed
```

4.3. SEQUENCING OF COMMANDS AND REPLIES

The communication between the sender and receiver is intended to be an alternating dialogue, controlled by the sender. As such, the sender issues a command and the receiver responds with a reply. The sender must wait for this response before sending further commands.

One important reply is the connection greeting. Normally, a receiver will send a 220 "Service ready" reply when the connection is completed. The sender should wait for this greeting message before sending any commands.

Note: all the greeting type replies have the official name of the server host as the first word following the reply code.

For example,

```
220 <SP> USC-ISIF.ARPA <SP> Service ready <CRLF>
```

The table below lists alternative success and failure replies for each command. These must be strictly adhered to; a receiver may substitute text in the replies, but the meaning and action implied by the code numbers and by the specific command reply sequence cannot be altered.

COMMAND-REPLY SEQUENCES

Each command is listed with its possible replies. The prefixes used before the possible replies are "P" for preliminary (not used in SMTP), "I" for intermediate, "S" for success, "F" for failure, and "E" for error. The 421 reply (service not available, closing transmission channel) may be given to any command if the SMTP-receiver knows it must shut down. This listing forms the basis for the State Diagrams in Section 4.4.

```
CONNECTION ESTABLISHMENT
   S: 220
   F: 421
HELO
   S: 250
   E: 500, 501, 504, 421
MAIL
   S: 250
   F: 552, 451, 452
   E: 500, 501, 421
```

```
            RCPT
               S: 250, 251
               F: 550, 551, 552, 553, 450, 451, 452
               E: 500, 501, 503, 421
            DATA
               I: 354 -> data -> S: 250
                                 F: 552, 554, 451, 452
               F: 451, 554
               E: 500, 501, 503, 421
            RSET
               S: 250
               E: 500, 501, 504, 421
            SEND
               S: 250
               F: 552, 451, 452
               E: 500, 501, 502, 421
            SOML
               S: 250
               F: 552, 451, 452
               E: 500, 501, 502, 421
            SAML
               S: 250
               F: 552, 451, 452
               E: 500, 501, 502, 421
            VRFY
               S: 250, 251
               F: 550, 551, 553
               E: 500, 501, 502, 504, 421
            EXPN
               S: 250
               F: 550
               E: 500, 501, 502, 504, 421
            HELP
               S: 211, 214
               E: 500, 501, 502, 504, 421
            NOOP
               S: 250
               E: 500, 421
            QUIT
               S: 221
               E: 500
            TURN
               S: 250
               F: 502
               E: 500, 503
```

4.4. STATE DIAGRAMS

Following are state diagrams for a simple-minded SMTP implementation. Only the first digit of the reply codes is used. There is one state diagram for each group of SMTP commands. The command groupings were determined by constructing a model for each command and then collecting together the commands with structurally identical models.

For each command there are three possible outcomes: "success" (S), "failure" (F), and "error" (E). In the state diagrams below we use the symbol B for "begin", and the symbol W for "wait for reply".

First, the diagram that represents most of the SMTP commands:

```
                                 1,3    +---+
                            ----------->| E |
                            |           +---+
                            |
       +---+    cmd    +---+    2       +---+
       | B |---------->| W |----------->| S |
       +---+           +---+            +---+
                            |
                            |    4,5    +---+
                            ----------->| F |
                                        +---+
```

This diagram models the commands:

HELO, MAIL, RCPT, RSET, SEND, SOML, SAML, VRFY, EXPN, HELP, NOOP, QUIT, TURN.

A more complex diagram models the DATA command:

```
  +---+   DATA    +---+ 1,2                 +---+
  | B |---------->| W |-------------------->| E |
  +---+           +---+        ------------>+---+
                  3| |4,5     |
                   | |        |
     --------------   -----   |
    |                      |  |             +---+
    |               ----------  ---------->| S |
    |              |       |   |            +---+
    |              |  ------------
    |              | |     |
    V           1,3| |2    |
  +---+   data    +---+     --------------->+---+
  |   |---------->| W |                     | F |
  +---+           +---+-------------------->+---+
                      4,5
```

Note that the "data" here is a series of lines sent from the sender to the receiver with no response expected until the last line is sent.

4.5. DETAILS

4.5.1. MINIMUM IMPLEMENTATION

In order to make SMTP workable, the following minimum implementation is required for all receivers:

```
COMMANDS -- HELO
            MAIL
            RCPT
            DATA
            RSET
            NOOP
            QUIT
```

4.5.2. TRANSPARENCY

Without some provision for data transparency the character sequence "<CRLF>.<CRLF>" ends the mail text and cannot be sent by the user. In general, users are not aware of such "forbidden" sequences. To allow all user composed text to be transmitted transparently the following procedures are used.

1. Before sending a line of mail text the sender-SMTP checks the first character of the line. If it is a period, one additional period is inserted at the beginning of the line.

2. When a line of mail text is received by the receiver-SMTP it checks the line. If the line is composed of a single period it is the end of mail. If the first character is a period and there are other characters on the line, the first character is deleted.

The mail data may contain any of the 128 ASCII characters. All characters are to be delivered to the recipient's mailbox including format effectors and other control characters. If the transmission channel provides an 8-bit byte (octets) data stream, the 7-bit ASCII codes are transmitted right justified in the octets with the high order bits cleared to zero.

In some systems it may be necessary to transform the data as it is received and stored. This may be necessary for hosts that use a different character set than ASCII as their local character set, or that store data in records rather than strings. If such transforms are necessary, they must be reversible -- especially if such transforms are applied to mail being relayed.

4.5.3. SIZES

There are several objects that have required minimum maximum sizes. That is, every implementation must be able to receive objects of at least these sizes, but must not send objects larger than these sizes.

```
****************************************************
*                                                  *
*  TO THE MAXIMUM EXTENT POSSIBLE, IMPLEMENTATION  *
*  TECHNIQUES WHICH IMPOSE NO LIMITS ON THE LENGTH *
*  OF THESE OBJECTS SHOULD BE USED.                *
*                                                  *
****************************************************
```

user

The maximum total length of a user name is 64 characters.

domain

The maximum total length of a domain name or number is 64 characters.

path

The maximum total length of a reverse-path or forward-path is 256 characters (including the punctuation and element separators).

command line

The maximum total length of a command line including the command word and the <CRLF> is 512 characters.

reply line

The maximum total length of a reply line including the reply code and the <CRLF> is 512 characters.

text line

The maximum total length of a text line including the <CRLF> is 1000 characters (but not counting the leading dot duplicated for transparency).

recipients buffer

The maximum total number of recipients that must be buffered is 100 recipients.

```
****************************************************
*                                                  *
*  TO THE MAXIMUM EXTENT POSSIBLE, IMPLEMENTATION  *
*  TECHNIQUES WHICH IMPOSE NO LIMITS ON THE LENGTH *
*  OF THESE OBJECTS SHOULD BE USED.                *
*                                                  *
****************************************************
```

Errors due to exceeding these limits may be reported by using the reply codes, for example:

500 Line too long.

501 Path too long

552 Too many recipients.

552 Too much mail data.

APPENDIX A

TCP Transport service

The Transmission Control Protocol [3] is used in the ARPA Internet, and in any network following the US DoD standards for internetwork protocols.

Connection Establishment

The SMTP transmission channel is a TCP connection established between the sender process port U and the receiver process port L. This single full duplex connection is used as the transmission channel. This protocol is assigned the service port 25 (31 octal), that is L=25.

Data Transfer

The TCP connection supports the transmission of 8-bit bytes. The SMTP data is 7-bit ASCII characters. Each character is transmitted as an 8-bit byte with the high-order bit cleared to zero.

APPENDIX B

NCP Transport service

The ARPANET Host-to-Host Protocol [4] (implemented by the Network Control Program) may be used in the ARPANET.

Connection Establishment

The SMTP transmission channel is established via NCP between the sender process socket U and receiver process socket L. The Initial Connection Protocol [5] is followed resulting in a pair of simplex connections. This pair of connections is used as the transmission channel. This protocol is assigned the contact socket 25 (31 octal), that is L=25.

Data Transfer

The NCP data connections are established in 8-bit byte mode. The SMTP data is 7-bit ASCII characters. Each character is transmitted as an 8-bit byte with the high-order bit cleared to zero.

APPENDIX C

NITS

The Network Independent Transport Service [6] may be used.

Connection Establishment

The SMTP transmission channel is established via NITS between the sender process and receiver process. The sender process executes the CONNECT primitive, and the waiting receiver process executes the ACCEPT primitive.

Data Transfer

The NITS connection supports the transmission of 8-bit bytes. The SMTP data is 7-bit ASCII characters. Each character is transmitted as an 8-bit byte with the high-order bit cleared to zero.

APPENDIX D

X.25 Transport service

It may be possible to use the X.25 service [7] as provided by the Public Data Networks directly, however, it is suggested that a reliable end-to-end protocol such as TCP be used on top of X.25 connections.

APPENDIX E

Theory of Reply Codes

The three digits of the reply each have a special significance. The first digit denotes whether the response is good, bad or incomplete. An unsophisticated sender-SMTP will be able to determine its next action (proceed as planned, redo, retrench, etc.) by simply examining this first digit. A sender-SMTP that wants to know approximately what kind of error occurred (e.g., mail system error, command syntax error) may examine the second digit, reserving the third digit for the finest gradation of information.

There are five values for the first digit of the reply code:

1yz Positive Preliminary reply

The command has been accepted, but the requestad action is being held in abeyance, pending confirmation of the information in this reply. The sender-SMTP should send another command specifying whether to continue or abort the action.

[Note: SMTP does not have any commands that allow this type of reply, and so does not have the continue or abort commands.]

2yz Positive Completion reply

The requested action has been successfully completed. A new request may be initiated.

3yz Positive Intermediate reply

The command has been accepted, but the requested action is being held in abeyance, pending receipt of further information. The sender-SMTP should send another command specifying this information. This reply is used in command sequence groups.

4yz Transient Negative Completion reply

The command was not accepted and the requested action did not occur. However, the error condition is temporary and the action may be requested again. The sender should return to the beginning of the command sequence (if any). It is difficult to assign a meaning to "transient" when two different sites (receiver- and sender- SMTPs) must agree on the interpretation. Each reply in this category might have a different time value, but the sender-SMTP is encouraged to try again. A rule of thumb to determine if a reply fits into the 4yz or the 5yz category (see below) is that replies are 4yz if they can be repeated without any change in command form or in properties of the sender or receiver. (E.g., the command is repeated identically and the receiver does not put up a new implementation.)

5yz Permanent Negative Completion reply

The command was not accepted and the requested action did not occur. The sender-SMTP is discouraged from repeating the exact request (in the same sequence). Even some "permanent" error conditions can be corrected, so the human user may want to direct the sender-SMTP to reinitiate the command sequence by direct action at some point in the future (e.g., after the spelling has been changed, or the user has altered the account status).

The second digit encodes responses in specific categories:

x0z Syntax -- These replies refer to syntax errors, syntactically correct commands that don't fit any functional category, and unimplemented or superfluous commands.

x1z Information -- These are replies to requests for information, such as status or help.

x2z Connections -- These are replies referring to the transmission channel.

x3z Unspecified as yet.

x4z Unspecified as yet.

x5z Mail system -- These replies indicate the status of the receiver mail system vis-a-vis the requested transfer or other mail system action.

The third digit gives a finer gradation of meaning in each category specified by the second digit. The list of replies illustrates this. Each reply text is recommended rather than mandatory, and may even change according to the command with which it is associated. On the other hand, the reply codes must strictly follow the specifications in this section. Receiver implementations should not invent new codes for slightly different situations from the ones described here, but rather adapt codes already defined.

For example, a command such as NOOP whose successful execution does not offer the sender-SMTP any new information will return a 250 reply. The response is 502 when the command requests an unimplemented non-site-specific action. A refinement of that is the 504 reply for a command that is implemented, but that requests an unimplemented parameter.

The reply text may be longer than a single line; in these cases the complete text must be marked so the sender-SMTP knows when it can stop reading the reply. This requires a special format to indicate a multiple line reply.

The format for multiline replies requires that every line, except the last, begin with the reply code, followed immediately by a hyphen, "-" (also known as minus), followed by text. The last line will begin with the reply code, followed immediately by <SP>, optionally some text, and <CRLF>.

For example:

```
123-First line
123-Second line
123-234 text beginning with numbers
123 The last line
```

In many cases the sender-SMTP then simply needs to search for the reply code followed by <SP> at the beginning of a line, and ignore all preceding lines. In a few cases, there is important data for the sender in the reply "text". The sender will know these cases from the current context.

APPENDIX F

Scenarios

This section presents complete scenarios of several types of SMTP sessions.

A Typical SMTP Transaction Scenario

This SMTP example shows mail sent by Smith at host USC-ISIF, to Jones, Green, and Brown at host BBN-UNIX. Here we assume that host USC-ISIF contacts host BBN-UNIX directly. The mail is accepted for Jones and Brown. Green does not have a mailbox at host BBN-UNIX.

```
R: 220 BBN-UNIX.ARPA Simple Mail Transfer Service Ready
S: HELO USC-ISIF.ARPA
R: 250 BBN-UNIX.ARPA

S: MAIL FROM:<Smith@USC-ISIF.ARPA>
R: 250 OK

S: RCPT TO:<Jones@BBN-UNIX.ARPA>
R: 250 OK

S: RCPT TO:<Green@BBN-UNIX.ARPA>
R: 550 No such user here

S: RCPT TO:<Brown@BBN-UNIX.ARPA>
R: 250 OK

S: DATA
R: 354 Start mail input; end with <CRLF>.<CRLF>
S: Blah blah blah...
S: ...etc. etc. etc.
S: .
R: 250 OK

S: QUIT
R: 221 BBN-UNIX.ARPA Service closing transmission channel
```

Scenario 1

Aborted SMTP Transaction Scenario

```
   -------------------------------------------------------------

      R: 220 MIT-Multics.ARPA Simple Mail Transfer Service Ready
      S: HELO ISI-VAXA.ARPA
      R: 250 MIT-Multics.ARPA

      S: MAIL FROM:<Smith@ISI-VAXA.ARPA>
      R: 250 OK

      S: RCPT TO:<Jones@MIT-Multics.ARPA>
      R: 250 OK

      S: RCPT TO:<Green@MIT-Multics.ARPA>
      R: 550 No such user here

      S: RSET
      R: 250 OK

      S: QUIT
      R: 221 MIT-Multics.ARPA Service closing transmission channel

                               Scenario 2

   -------------------------------------------------------------
```

Relayed Mail Scenario

```
   -------------------------------------------------------------

      Step 1 --  Source Host to Relay Host

         R: 220 USC-ISIE.ARPA Simple Mail Transfer Service Ready
         S: HELO MIT-AI.ARPA
         R: 250 USC-ISIE.ARPA

         S: MAIL FROM:<JQP@MIT-AI.ARPA>
         R: 250 OK

         S: RCPT TO:<@USC-ISIE.ARPA:Jones@BBN-VAX.ARPA>
         R: 250 OK

         S: DATA
         R: 354 Start mail input; end with <CRLF>.<CRLF>
         S: Date: 2 Nov 81 22:33:44
         S: From: John Q. Public <JQP@MIT-AI.ARPA>
         S: Subject:  The Next Meeting of the Board
         S: To: Jones@BBN-Vax.ARPA
         S:
         S: Bill:
         S: The next meeting of the board of directors will be
         S: on Tuesday.
         S:                                                  John.
         S: .
         R: 250 OK

         S: QUIT
         R: 221 USC-ISIE.ARPA Service closing transmission channel
      Step 2 -- Relay Host to Destination Host

         R: 220 BBN-VAX.ARPA Simple Mail Transfer Service Ready
         S: HELO USC-ISIE.ARPA
         R: 250 BBN-VAX.ARPA

         S: MAIL FROM:<@USC-ISIE.ARPA:JQP@MIT-AI.ARPA>
         R: 250 OK

         S: RCPT TO:<Jones@BBN-VAX.ARPA>
         R: 250 OK

         S: DATA
         R: 354 Start mail input; end with <CRLF>.<CRLF>
         S: Received: from MIT-AI.ARPA by USC-ISIE.ARPA ;
            2 Nov 81 22:40:10 UT
         S; Date: 2 Nov 81 22:33:44
         S: From: John Q. Public <JQP@MIT-AI.ARPA>
         S: Subject:  The Next Meeting of the Board
         S: To: Jones@BBN-Vax.ARPA
         S:
         S: Bill:
         S: The next meeting of the board of directors will be
         S: on Tuesday.
         S:                                             John.
         S: .
         R: 250 OK

         S: QUIT
         R: 221 USC-ISIE.ARPA Service closing transmission channel

                               Scenario 3

   -------------------------------------------------------------
```

Verifying and Sending Scenario

```
R: 220 SU-SCORE.ARPA Simple Mail Transfer Service Ready
S: HELO MIT-MC.ARPA
R: 250 SU-SCORE.ARPA

S: VRFY Crispin
R: 250 Mark Crispin <Admin.MRC@SU-SCORE.ARPA>

S: SEND FROM:<EAK@MIT-MC.ARPA>
R: 250 OK

S: RCPT TO:<Admin.MRC@SU-SCORE.ARPA>
R: 250 OK

S: DATA
R: 354 Start mail input; end with <CRLF>.<CRLF>
S: Blah blah blah...
S: ...etc. etc. etc.
S: .
R: 250 OK

S: QUIT
R: 221 SU-SCORE.ARPA Service closing transmission channel
```

Scenario 4

Sending and Mailing Scenarios

First the user's name is verified, then an attempt is made to send to the user's terminal. When that fails, the messages is mailed to the user's mailbox.

```
R: 220 SU-SCORE.ARPA Simple Mail Transfer Service Ready
S: HELO MIT-MC.ARPA
R: 250 SU-SCORE.ARPA

S: VRFY Crispin
R: 250 Mark Crispin <Admin.MRC@SU-SCORE.ARPA>

S: SEND FROM:<EAK@MIT-MC.ARPA>
R: 250 OK

S: RCPT TO:<Admin.MRC@SU-SCORE.ARPA>
R: 450 User not active now

S: RSET
R: 250 OK

S: MAIL FROM:<EAK@MIT-MC.ARPA>
R: 250 OK

S: RCPT TO:<Admin.MRC@SU-SCORE.ARPA>
R: 250 OK

S: DATA
R: 354 Start mail input; end with <CRLF>.<CRLF>
S: Blah blah blah...
S: ...etc. etc. etc.
S: .
R: 250 OK

S: QUIT
R: 221 SU-SCORE.ARPA Service closing transmission channel
```

Scenario 5

Doing the preceding scenario more efficiently.

```
-------------------------------------------------------------

      R: 220 SU-SCORE.ARPA Simple Mail Transfer Service Ready
      S: HELO MIT-MC.ARPA
      R: 250 SU-SCORE.ARPA

      S: VRFY Crispin
      R: 250 Mark Crispin <Admin.MRC@SU-SCORE.ARPA>

      S: SOML FROM:<EAK@MIT-MC.ARPA>
      R: 250 OK

      S: RCPT TO:<Admin.MRC@SU-SCORE.ARPA>
      R: 250 User not active now, so will do mail.

      S: DATA
      R: 354 Start mail input; end with <CRLF>.<CRLF>
      S: Blah blah blah...
      S: ...etc. etc. etc.
      S: .
      R: 250 OK

      S: QUIT
      R: 221 SU-SCORE.ARPA Service closing transmission channel

                           Scenario 6

-------------------------------------------------------------
```

Mailing List Scenario

First each of two mailing lists are expanded in separate sessions with different hosts. Then the message is sent to everyone that appeared on either list (but no duplicates) via a relay host.

```
   Step 1  --  Expanding the First List

      R: 220 MIT-AI.ARPA Simple Mail Transfer Service Ready
      S: HELO SU-SCORE.ARPA
      R: 250 MIT-AI.ARPA

      S: EXPN Example-People
      R: 250-<ABC@MIT-MC.ARPA>
      R: 250-Fred Fonebone <Fonebone@USC-ISIQ.ARPA>
      R: 250-Xenon Y. Zither <XYZ@MIT-AI.ARPA>
      R: 250-Quincy Smith <@USC-ISIF.ARPA:Q-Smith@ISI-VAXA.ARPA>
      R: 250-<joe@foo-unix.ARPA>
      R: 250 <xyz@bar-unix.ARPA>

      S: QUIT
      R: 221 MIT-AI.ARPA Service closing transmission channel
```

Step 2 -- Expanding the Second List

```
R: 220 MIT-MC.ARPA Simple Mail Transfer Service Ready
S: HELO SU-SCORE.ARPA
R: 250 MIT-MC.ARPA

S: EXPN Interested-Parties
R: 250-Al Calico <ABC@MIT-MC.ARPA>
R: 250-<XYZ@MIT-AI.ARPA>
R: 250-Quincy Smith <@USC-ISIF.ARPA:Q-Smith@ISI-VAXA.ARPA>
R: 250-<fred@BBN-UNIX.ARPA>
R: 250 <xyz@bar-unix.ARPA>

S: QUIT
R: 221 MIT-MC.ARPA Service closing transmission channel
```

Step 3 -- Mailing to All via a Relay Host

```
R: 220 USC-ISIE.ARPA Simple Mail Transfer Service Ready
S: HELO SU-SCORE.ARPA
R: 250 USC-ISIE.ARPA

S: MAIL FROM:<Account.Person@SU-SCORE.ARPA>
R: 250 OK
S: RCPT TO:<@USC-ISIE.ARPA:ABC@MIT-MC.ARPA>
R: 250 OK
S: RCPT TO:<@USC-ISIE.ARPA:Fonebone@USC-ISIQA.ARPA>
R: 250 OK
S: RCPT TO:<@USC-ISIE.ARPA:XYZ@MIT-AI.ARPA>
R: 250 OK
S: RCPT
    TO:<@USC-ISIE.ARPA,@USC-ISIF.ARPA:Q-Smith@ISI-VAXA.ARPA>
R: 250 OK
S: RCPT TO:<@USC-ISIE.ARPA:joe@FOO-UNIX.ARPA>
R: 250 OK
S: RCPT TO:<@USC-ISIE.ARPA:xyz@BAR-UNIX.ARPA>
R: 250 OK
S: RCPT TO:<@USC-ISIE.ARPA:fred@BBN-UNIX.ARPA>
R: 250 OK

S: DATA
R: 354 Start mail input; end with <CRLF>.<CRLF>
S: Blah blah blah...
S: ...etc. etc. etc.
S: .
R: 250 OK

S: QUIT
R: 221 USC-ISIE.ARPA Service closing transmission channel
```

Scenario 7

Forwarding Scenarios

```
      -------------------------------------------------------------

         R: 220 USC-ISIF.ARPA Simple Mail Transfer Service Ready
         S: HELO LBL-UNIX.ARPA
         R: 250 USC-ISIF.ARPA

         S: MAIL FROM:<mo@LBL-UNIX.ARPA>
         R: 250 OK

         S: RCPT TO:<fred@USC-ISIF.ARPA>
         R: 251 User not local; will forward to <Jones@USC-ISI.ARPA>

         S: DATA
         R: 354 Start mail input; end with <CRLF>.<CRLF>
         S: Blah blah blah...
         S: ...etc. etc. etc.
         S: .
         R: 250 OK

         S: QUIT
         R: 221 USC-ISIF.ARPA Service closing transmission channel

                                 Scenario 8

      -------------------------------------------------------------
```

```
-------------------------------------------------------------

   Step 1  --  Trying the Mailbox at the First Host

      R: 220 USC-ISIF.ARPA Simple Mail Transfer Service Ready
      S: HELO LBL-UNIX.ARPA
      R: 250 USC-ISIF.ARPA

      S: MAIL FROM:<mo@LBL-UNIX.ARPA>
      R: 250 OK

      S: RCPT TO:<fred@USC-ISIF.ARPA>
      R: 251 User not local; will forward to <Jones@USC-ISI.ARPA>

      S: RSET
      R: 250 OK

      S: QUIT
      R: 221 USC-ISIF.ARPA Service closing transmission channel

   Step 2  --  Delivering the Mail at the Second Host

      R: 220 USC-ISI.ARPA Simple Mail Transfer Service Ready
      S: HELO LBL-UNIX.ARPA
      R: 250 USC-ISI.ARPA

      S: MAIL FROM:<mo@LBL-UNIX.ARPA>
      R: 250 OK

      S: RCPT TO:<Jones@USC-ISI.ARPA>
      R: OK

      S: DATA
      R: 354 Start mail input; end with <CRLF>.<CRLF>
      S: Blah blah blah...
      S: ...etc. etc. etc.
      S: .
      R: 250 OK

      S: QUIT
      R: 221 USC-ISI.ARPA Service closing transmission channel
```

Scenario 9

```
-------------------------------------------------------------
```

Too Many Recipients Scenario

```
R: 220 BERKELEY.ARPA Simple Mail Transfer Service Ready
S: HELO USC-ISIF.ARPA
R: 250 BERKELEY.ARPA

S: MAIL FROM:<Postel@USC-ISIF.ARPA>
R: 250 OK

S: RCPT TO:<fabry@BERKELEY.ARPA>
R: 250 OK

S: RCPT TO:<eric@BERKELEY.ARPA>
R: 552 Recipient storage full, try again in another transaction

S: DATA
R: 354 Start mail input; end with <CRLF>.<CRLF>
S: Blah blah blah...
S: ...etc. etc. etc.
S: .
R: 250 OK

S: MAIL FROM:<Postel@USC-ISIF.ARPA>
R: 250 OK

S: RCPT TO:<eric@BERKELEY.ARPA>
R: 250 OK

S: DATA
R: 354 Start mail input; end with <CRLF>.<CRLF>
S: Blah blah blah...
S: ...etc. etc. etc.
S: .
R: 250 OK

S: QUIT
R: 221 BERKELEY.ARPA Service closing transmission channel
```

Scenario 10

Note that a real implementation must handle many recipients as specified in Section 4.5.3.

GLOSSARY

ASCII

American Standard Code for Information Interchange [1].

command

A request for a mail service action sent by the sender-SMTP to the receiver-SMTP.

domain

The hierarchially structured global character string address of a host computer in the mail system.

end of mail data indication

A special sequence of characters that indicates the end of the mail data. In particular, the five characters carriage return, line feed, period, carriage return, line feed, in that order.

host

A computer in the internetwork environment on which mailboxes or SMTP processes reside.

line

A a sequence of ASCII characters ending with a <CRLF>.

mail data

A sequence of ASCII characters of arbitrary length, which conforms to the standard set in the Standard for the Format of ARPA Internet Text Messages (RFC 822 [2]).

mailbox

A character string (address) which identifies a user to whom mail is to be sent. Mailbox normally consists of the host and user specifications. The standard mailbox naming convention is defined to be "user@domain". Additionally, the "container" in which mail is stored.

receiver-SMTP process

A process which transfers mail in cooperation with a sender-SMTP process. It waits for a connection to be established via the transport service. It receives SMTP commands from the sender-SMTP, sends replies, and performs the specified operations.

reply

A reply is an acknowledgment (positive or negative) sent from receiver to sender via the transmission channel in response to a command. The general form of a reply is a completion code (including error codes) followed by a text string. The codes are for use by programs and the text is usually intended for human users.

sender-SMTP process

A process which transfers mail in cooperation with a receiver-SMTP process. A local language may be used in the user interface command/reply dialogue. The sender-SMTP initiates the transport service connection. It initiates SMTP commands, receives replies, and governs the transfer of mail.

session

The set of exchanges that occur while the transmission channel is open.

transaction

The set of exchanges required for one message to be transmitted for one or more recipients.

transmission channel

A full-duplex communication path between a sender-SMTP and a receiver-SMTP for the exchange of commands, replies, and mail text.

transport service

Any reliable stream-oriented data communication services. For example, NCP, TCP, NITS.

user

A human being (or a process on behalf of a human being) wishing to obtain mail transfer service. In addition, a recipient of computer mail.

word

A sequence of printing characters.

<CRLF>

The characters carriage return and line feed (in that order).

<SP>

The space character.

REFERENCES

[1] ASCII

ASCII, "USA Codo for Information Interchange", United States of America Standards Institute, X3.4, 1968. Also in: Feinler, E. and J. Postel, eds., "ARPANET Protocol Handbook", NIC 7104, for the Defense Communications Agency by SRI International, Menlo Park, California, Revised January 1978.

[2] RFC 822

Crocker, D., "Standard for the Format of ARPA Internet Text Messages," RFC 822, Department of Electrical Engineering, University of Delaware, August 1982.

[3] TCP

Postel, J., ed., "Transmission Control Protocol - DARPA Internet Program Protocol Specification", RFC 793, USC/Information Sciences Institute, NTIS AD Number A111091, September 1981. Also in: Feinler, E. and J. Postel, eds., "Internet Protocol Transition Workbook", SRI International, Menlo Park, California, March 1982.

[4] NCP

McKenzie,A., "Host/Host Protocol for the ARPA Network", NIC 8246, January 1972. Also in: Feinler, E. and J. Postel, eds., "ARPANET Protocol Handbook", NIC 7104, for the Defense Communications Agency by SRI International, Menlo Park, California, Revised January 1978.

[5] Initial Connection Protocol

Postel, J., "Official Initial Connection Protocol", NIC 7101, 11 June 1971. Also in: Feinler, E. and J. Postel, eds., "ARPANET Protocol Handbook", NIC 7104, for the Defense Communications Agency by SRI International, Menlo Park, California, Revised January 1978.

[6] NITS

PSS/SG3, "A Network Independent Transport Service", Study Group 3, The Post Office PSS Users Group, February 1980. Available from the DCPU, National Physical Laboratory, Teddington, UK.

[7] X.25

CCITT, "Recommendation X.25 - Interface Between Data Terminal Equipment (DTE) and Data Circuit-terminating Equipment (DCE) for Terminals Operating in the Packet Mode on Public Data Networks," CCITT Orange Book, Vol. VIII.2, International Telephone and Telegraph Consultative Committee, Geneva, 1976.

RFC # 822

Obsoletes: RFC #733 (NIC #41952)

STANDARD FOR THE FORMAT OF ARPA INTERNET TEXT MESSAGES

August 13, 1982

Revised by

David H. Crocker

Dept. of Electrical Engineering
University of Delaware, Newark, DE 19711
Network: DCrocker @ UDel-Relay

TABLE OF CONTENTS

PREFACE ii

1. INTRODUCTION 1

   1.1. Scope 1
   1.2. Communication Framework 2

2. NOTATIONAL CONVENTIONS 3

3. LEXICAL ANALYSIS OF MESSAGES 5

   3.1. General Description 5
   3.2. Header Field Definitions 9
   3.3. Lexical Tokens 10
   3.4. Clarifications 11

4. MESSAGE SPECIFICATION 17

   4.1. Syntax 17
   4.2. Forwarding 19
   4.3. Trace Fields 20
   4.4. Originator Fields 21
   4.5. Receiver Fields 23
   4.6. Reference Fields 23
   4.7. Other Fields 24

5. DATE AND TIME SPECIFICATION 26

   5.1. Syntax 26
   5.2. Semantics 26

6. ADDRESS SPECIFICATION 27

   6.1. Syntax 27
   6.2. Semantics 27
   6.3. Reserved Address 33

7. BIBLIOGRAPHY 34

APPENDIX

A. EXAMPLES 36
B. SIMPLE FIELD PARSING 40
C. DIFFERENCES FROM RFC #733 41
D. ALPHABETICAL LISTING OF SYNTAX RULES 44

PREFACE

By 1977, the Arpanet employed several informal standards for the text messages (mail) sent among its host computers. It was felt necessary to codify these practices and provide for those features that seemed imminent. The result of that effort was Request for Comments (RFC) #733, "Standard for the Format of ARPA Network Text Message", by Crocker, Vittal, Pogran, and Henderson. The specification attempted to avoid major changes in existing software, while permitting several new features.

This document revises the specifications in RFC #733, in order to serve the needs of the larger and more complex ARPA Internet. Some of RFC #733's features failed to gain adequate acceptance. In order to simplify the standard and the software that follows it, these features have been removed. A different addressing scheme is used, to handle the case of inter-network mail; and the concept of re-transmission has been introduced.

This specification is intended for use in the ARPA Internet. However, an attempt has been made to free it of any dependence on that environment, so that it can be applied to other network text message systems.

The specification of RFC #733 took place over the course of one year, using the ARPANET mail environment, itself, to provide an on-going forum for discussing the capabilities to be included. More than twenty individuals, from across the country, participated in the original discussion. The development of this revised specification has, similarly, utilized network mail-based group discussion. Both specification efforts greatly benefited from the comments and ideas of the participants.

The syntax of the standard, in RFC #733, was originally specified in the Backus-Naur Form (BNF) meta-language. Ken L. Harrenstien, of SRI International, was responsible for re-coding the BNF into an augmented BNF that makes the representation smaller and easier to understand.

1. INTRODUCTION

1.1. SCOPE

This standard specifies a syntax for text messages that are sent among computer users, within the framework of "electronic mail". The standard supersedes the one specified in ARPANET Request for Comments #733, "Standard for the Format of ARPA Network Text Messages".

In this context, messages are viewed as having an envelope and contents. The envelope contains whatever information is needed to accomplish transmission and delivery. The contents compose the object to be delivered to the recipient. This standard applies only to the format and some of the semantics of message contents. It contains no specification of the information in the envelope.

However, some message systems may use information from the contents to create the envelope. It is intended that this standard facilitate the acquisition of such information by programs.

Some message systems may store messages in formats that differ from the one specified in this standard. This specification is intended strictly as a definition of what message content format is to be passed BETWEEN hosts.

Note: This standard is NOT intended to dictate the internal formats used by sites, the specific message system features that they are expected to support, or any of the characteristics of user interface programs that create or read messages.

A distinction should be made between what the specification REQUIRES and what it ALLOWS. Messages can be made complex and rich with formally-structured components of information or can be kept small and simple, with a minimum of such information. Also, the standard simplifies the interpretation of differing visual formats in messages; only the visual aspect of a message is affected and not the interpretation of information within it. Implementors may choose to retain such visual distinctions.

The formal definition is divided into four levels. The bottom level describes the meta-notation used in this document. The second level describes basic lexical analyzers that feed tokens to higher-level parsers. Next is an overall specification for messages; it permits distinguishing individual fields. Finally, there is definition of the contents of several structured fields.

1.2. COMMUNICATION FRAMEWORK

Messages consist of lines of text. No special provisions are made for encoding drawings, facsimile, speech, or structured text. No significant consideration has been given to questions of data compression or to transmission and storage efficiency, and the standard tends to be free with the number of bits consumed. For example, field names are specified as free text, rather than special terse codes.

A general "memo" framework is used. That is, a message consists of some information in a rigid format, followed by the main part of the message, with a format that is not specified in this document. The syntax of several fields of the rigidly-formated ("headers") section is defined in this specification; some of these fields must be included in all messages.

The syntax that distinguishes between header fields is specified separately from the internal syntax for particular fields. This separation is intended to allow simple parsers to operate on the general structure of messages, without concern for the detailed structure of individual header fields. Appendix B is provided to facilitate construction of these parsers.

In addition to the fields specified in this document, it is expected that other fields will gain common use. As necessary, the specifications for these "extension-fields" will be published through the same mechanism used to publish this document. Users may also wish to extend the set of fields that they use privately. Such "user-defined fields" are permitted.

The framework severely constrains document tone and appearance and is primarily useful for most intra-organization communications and well-structured inter-organization communication. It also can be used for some types of inter-process communication, such as simple file transfer and remote job entry. A more robust framework might allow for multi-font, multi-color, multi-dimension encoding of information. A less robust one, as is present in most single-machine message systems, would more severely constrain the ability to add fields and the decision to include specific fields. In contrast with paper-based communication, it is interesting to note that the RECEIVER of a message can exercise an extraordinary amount of control over the message's appearance. The amount of actual control available to message receivers is contingent upon the capabilities of their individual message systems.

2. NOTATIONAL CONVENTIONS

This specification uses an augmented Backus-Naur Form (BNF) notation. The differences from standard BNF involve naming rules and indicating repetition and "local" alternatives.

2.1. RULE NAMING

Angle brackets ("<", ">") are not used, in general. The name of a rule is simply the name itself, rather than "<name>". Quotation-marks enclose literal text (which may be upper and/or lower case). Certain basic rules are in uppercase, such as SPACE, TAP, CRLF, DIGIT, ALPHA, etc. Angle brackets are used in rule definitions, and in the rest of this document, whenever their presence will facilitate discerning the use of rule names.

2.2. RULE1 / RULE2: ALTERNATIVES

Elements separated by slash ("/") are alternatives. Therefore "foo / bar" will accept foo or bar.

2.3. (RULE1 RULE2): LOCAL ALTERNATIVES

Elements enclosed in parentheses are treated as a single element. Thus, "(elem (foo / bar) elem)" allows the token sequences "elem foo elem" and "elem bar elem".

2.4. *RULE: REPETITION

The character "*" preceding an element indicates repetition. The full form is:

<l>*<m>element

indicating at least <l> and at most <m> occurrences of element. Default values are 0 and infinity so that "*(element)" allows any number, including zero; "1*element" requires at least one; and "1*2element" allows one or two.

2.5. [RULE]: OPTIONAL

Square brackets enclose optional elements; "[foo bar]" is equivalent to "*1(foo bar)".

2.6. NRULE: SPECIFIC REPETITION

"<n>(element)" is equivalent to "<n>*<n>(element)"; that is, exactly <n> occurrences of (element). Thus 2DIGIT is a 2-digit number, and 3ALPHA is a string of three alphabetic characters.

2.7. #RULE: LISTS

A construct "#" is defined, similar to "*", as follows:

<1>#<m>element

indicating at least <1> and at most <m> elements, each separated by one or more commas (","). This makes the usual form of lists very easy; a rule such as '(element *("," element))' can be shown as "1#element". Wherever this construct is used, null elements are allowed, but do not contribute to the count of elements present. That is, "(element),,(element)" is permitted, but counts as only two elements. Therefore, where at least one element is required, at least one non-null element must be present. Default values are 0 and infinity so that "#(element)" allows any number, including zero; "1#element" requires at least one; and "1#2element" allows one or two.

2.8. ; COMMENTS

A semi-colon, set off some distance to the right of rule text, starts a comment that continues to the end of line. This is a simple way of including useful notes in parallel with the specifications.

3. LEXICAL ANALYSIS OF MESSAGES

3.1. GENERAL DESCRIPTION

A message consists of header fields and, optionally, a body. The body is simply a sequence of lines containing ASCII characters. It is separated from the headers by a null line (i.e., a line with nothing preceding the CRLF).

3.1.1. LONG HEADER FIELDS

Each header field can be viewed as a single, logical line of ASCII characters, comprising a field-name and a field-body. For convenience, the field-body portion of this conceptual entity can be split into a multiple-line representation; this is called "folding". The general rule is that wherever there may be linear-white-space (NOT simply LWSP-chars), a CRLF immediately followed by AT LEAST one LWSP-char may instead be inserted. Thus, the single line

```
To:  "Joe & J. Harvey" <ddd @Org>, JJV @ BBN
```

can be represented as:

```
To:  "Joe & J. Harvey" <ddd @ Org>,
         JJV@BBN
```

and

```
To:  "Joe & J. Harvey"
                 <ddd@ Org>, JJV
 @BBN
```

and

```
To:  "Joe &
 J. Harvey" <ddd @ Org>, JJV @ BBN
```

The process of moving from this folded multiple-line representation of a header field to its single line representation is called "unfolding". Unfolding is accomplished by regarding CRLF immediately followed by a LWSP-char as equivalent to the LWSP-char.

Note: While the standard permits folding wherever linear-white-space is permitted, it is recommended that structured fields, such as those containing addresses, limit folding to higher-level syntactic breaks. For address fields, it is recommended that such folding occur between addresses, after the separating comma.

3.1.2. STRUCTURE OF HEADER FIELDS

Once a field has been unfolded, it may be viewed as being composed of a field-name followed by a colon (":"), followed by a field-body, and terminated by a carriage-return/line-feed. The field-name must be composed of printable ASCII characters (i.e., characters that have values between 33. and 126., decimal, except colon). The field-body may be composed of any ASCII characters, except CR or LF. (While CR and/or LF may be present in the actual text, they are removed by the action of unfolding the field.)

Certain field-bodies of headers may be interpreted according to an internal syntax that some systems may wish to parse. These fields are called "structured fields". Examples include fields containing dates and addresses. Other fields, such as "Subject" and "Comments", are regarded simply as strings of text.

Note: Any field which has a field-body that is defined as other than simply <text> is to be treated as a structured field.

Field-names, unstructured field bodies and structured field bodies each are scanned by their own, independent "lexical" analyzers.

3.1.3. UNSTRUCTURED FIELD BODIES

For some fields, such as "Subject" and "Comments", no structuring is assumed, and they are treated simply as <text>s, as in the message body. Rules of folding apply to these fields, so that such field bodies which occupy several lines must therefore have the second and successive lines indented by at least one LWSP-char.

3.1.4. STRUCTURED FIELD BODIES

To aid in the creation and reading of structured fields, the free insertion of linear-white-space (which permits folding by inclusion of CRLFs) is allowed between lexical tokens. Rather than obscuring the syntax specifications for these structured fields with explicit syntax for this linear-white-space, the existence of another "lexical" analyzer is assumed. This analyzer does not apply for unstructured field bodies that are simply strings of text, as described above. The analyzer provides an interpretation of the unfolded text composing the body of the field as a sequence of lexical symbols.

These symbols are:

- individual special characters
- quoted-strings
- domain-literals
- comments
- atoms

The first four of these symbols are self-delimiting. Atoms are not; they are delimited by the self-delimiting symbols and by linear-white-space. For the purposes of regenerating sequences of atoms and quoted-strings, exactly one SPACE is assumed to exist, and should be used, between them. (Also, in the "Clarifications" section on "White Space", below, note the rules about treatment of multiple contiguous LWSP-chars.)

So, for example, the folded body of an address field

```
    ":sysmail"@  Some-Group. Some-Org,
    Muhammed.(I am  the greatest) Ali @(the)Vegas.WBA
```

is analyzed into the following lexical symbols and types:

:sysmail	quoted string
@	special
Some-Group	atom
.	special
Some-Org	atom
.	special
Muhammed	atom
.	special
(I am the greatest)	comment
Ali	atom
@	atom
(the)	comment
Vegas	atom
.	special
WBA	atom

The canonical representations for the data in these addresses are the following strings:

":sysmail"@Some-Group.Some-Org

and

Muhammed.Ali@Vegas.WBA

Note: For purposes of display, and when passing such structured information to other systems, such as mail protocol services, there must be NO linear-white-space between <word>s that are separated by period (".") or at-sign ("@") and exactly one SPACE between all other <word>s. Also, headers should be in a folded form.

3.2. HEADER FIELD DEFINITIONS

These rules show a field meta-syntax, without regard for the particular type or internal syntax. Their purpose is to permit detection of fields; also, they present to higher-level parsers an image of each field as fitting on one line.

```
field       =  field-name ":" [ field-body ] CRLF

field-name  =  1*<any CHAR, excluding CTLs, SPACE, and ":">

field-body  =  field-body-contents
               [CRLF LWSP-char field-body]

field-body-contents =
              <the ASCII characters making up the field-body, as
               defined in the following sections, and consisting
               of combinations of atom, quoted-string, and
               specials tokens, or else consisting of texts>
```

3.3. LEXICAL TOKENS

The following rules are used to define an underlying lexical analyzer, which feeds tokens to higher level parsers. See the ANSI references, in the Bibliography.

```
                                                 ; (  Octal, Decimal.)
CHAR        =  <any ASCII character>             ; (  0-177,  0.-127.)
ALPHA       =  <any ASCII alphabetic character>
                                                 ; (101-132, 65.- 90.)
                                                 ; (141-172, 97.-122.)
DIGIT       =  <any ASCII decimal digit>         ; ( 60- 71, 48.- 57.)
CTL         =  <any ASCII control                ; (  0- 37,  0.- 31.)
                character and DEL>               ; (    177,     127.)
CR          =  <ASCII CR, carriage return>       ; (     15,      13.)
LF          =  <ASCII LF, linefeed>              ; (     12,      10.)
SPACE       =  <ASCII SP, space>                 ; (     40,      32.)
HTAB        =  <ASCII HT, horizontal-tab>        ; (     11,       9.)
<">         =  <ASCII quote mark>                ; (     42,      34.)
CRLF        =  CR LF

LWSP-char   =  SPACE / HTAB                      ; semantics = SPACE

linear-white-space =  1*([CRLF] LWSP-char)       ; semantics = SPACE
                                                 ; CRLF => folding

specials    =  "(" / ")" / "<" / ">" / "@"       ; Must be in quoted-
            /  "," / ";" / ":" / "\" / <">       ;  string, to use
            /  "." / "[" / "]"                   ;  within a word.

delimiters  =  specials / linear-white-space / comment

text        =  <any CHAR, including bare         ; => atoms, specials,
                CR & bare LF, but NOT            ;  comments and
                including CRLF>                  ;  quoted-strings are
                                                 ;  NOT recognized.

atom        =  1*<any CHAR except specials, SPACE and CTLs>

quoted-string = <"> *(qtext/quoted-pair) <">; Regular qtext or
                                                 ;   quoted chars.

qtext       =  <any CHAR excepting <">,          ; => may be folded
                "\" & CR, and including
                linear-white-space>

domain-literal =  "[" *(dtext / quoted-pair) "]"
```

```
dtext       =  <any CHAR excluding "[",     ; => may be folded
                "]", "\" & CR, & including
                linear-white-space>

comment     =  "(" *(ctext / quoted-pair / comment) ")"

ctext       =  <any CHAR excluding "(",     ; => may be folded
                ")", "\" & CR, & including
                linear-white-space>

quoted-pair =  "\" CHAR                     ; may quote any char

phrase      =  1*word                       ; Sequence of words

word        =  atom / quoted-string
```

3.4. CLARIFICATIONS

3.4.1. QUOTING

Some characters are reserved for special interpretation, such as delimiting lexical tokens. To permit use of these characters as uninterpreted data, a quoting mechanism is provided. To quote a character, precede it with a backslash ("\").

This mechanism is not fully general. Characters may be quoted only within a subset of the lexical constructs. In particular, quoting is limited to use within:

- quoted-string
- domain-literal
- comment

Within these constructs, quoting is REQUIRED for CR and "\" and for the character(s) that delimit the token (e.g., "(" and ")" for a comment). However, quoting is PERMITTED for any character.

Note: In particular, quoting is NOT permitted within atoms. For example when the local-part of an addr-spec must contain a special character, a quoted string must be used. Therefore, a specification such as:

```
Full\ Name@Domain
```

is not legal and must be specified as:

```
"Full Name"@Domain
```

3.4.2. WHITE SPACE

Note: In structured field bodies, multiple linear space ASCII characters (namely HTABs and SPACEs) are treated as single spaces and may freely surround any symbol. In all header fields, the only place in which at least one LWSP-char is REQUIRED is at the beginning of continuation lines in a folded field.

When passing text to processes that do not interpret text according to this standard (e.g., mail protocol servers), then NO linear-white-space characters should occur between a period (".") or at-sign ("@") and a <word>. Exactly ONE SPACE should be used in place of arbitrary linear-white-space and comment sequences.

Note: Within systems conforming to this standard, wherever a member of the list of delimiters is allowed, LWSP-chars may also occur before and/or after it.

Writers of mail-sending (i.e., header-generating) programs should realize that there is no network-wide definition of the effect of ASCII HT (horizontal-tab) characters on the appearance of text at another network host; therefore, the use of tabs in message headers, though permitted, is discouraged.

3.4.3. COMMENTS

A comment is a set of ASCII characters, which is enclosed in matching parentheses and which is not within a quoted-string The comment construct permits message originators to add text which will be useful for human readers, but which will be ignored by the formal semantics. Comments should be retained while the message is subject to interpretation according to this standard. However, comments must NOT be included in other cases, such as during protocol exchanges with mail servers.

Comments nest, so that if an unquoted left parenthesis occurs in a comment string, there must also be a matching right parenthesis. When a comment acts as the delimiter between a sequence of two lexical symbols, such as two atoms, it is lexically equivalent with a single SPACE, for the purposes of regenerating the sequence, such as when passing the sequence onto a mail protocol server. Comments are detected as such only within field-bodies of structured fields.

If a comment is to be "folded" onto multiple lines, then the syntax for folding must be adhered to. (See the "Lexical

Analysis of Messages" section on "Folding Long Header Fields" above, and the section on "Case Independence" below.) Note that the official semantics therefore do not "see" any unquoted CRLFs that are in comments, although particular parsing programs may wish to note their presence. For these programs, it would be reasonable to interpret a "CRLF LWSP-char" as being a CRLF that is part of the comment; i.e., the CRLF is kept and the LWSP-char is discarded. Quoted CRLFs (i.e., a backslash followed by a CR followed by a LF) still must be followed by at least one LWSP-char.

3.4.4. DELIMITING AND QUOTING CHARACTERS

The quote character (backslash) and characters that delimit syntactic units are not, generally, to be taken as data that are part of the delimited or quoted unit(s). In particular, the quotation-marks that define a quoted-string, the parentheses that define a comment and the backslash that quotes a following character are NOT part of the quoted-string, comment or quoted character. A quotation-mark that is to be part of a quoted-string, a parenthesis that is to be part of a comment and a backslash that is to be part of either must each be preceded by the quote-character backslash ("\"). Note that the syntax allows any character to be quoted within a quoted-string or comment; however only certain characters MUST be quoted to be included as data. These characters are the ones that are not part of the alternate text group (i.e., ctext or qtext).

The one exception to this rule is that a single SPACE is assumed to exist between contiguous words in a phrase, and this interpretation is independent of the actual number of LWSP-chars that the creator places between the words. To include more than one SPACE, the creator must make the LWSP-chars be part of a quoted-string.

Quotation marks that delimit a quoted string and backslashes that quote the following character should NOT accompany the quoted-string when the string is passed to processes that do not interpret data according to this specification (e.g., mail protocol servers).

3.4.5. QUOTED-STRINGS

Where permitted (i.e., in words in structured fields) quoted-strings are treated as a single symbol. That is, a quoted-string is equivalent to an atom, syntactically. If a quoted-string is to be "folded" onto multiple lines, then the syntax for folding must be adhered to. (See the "Lexical Analysis of

Messages" section on "Folding Long Header Fields" above, and the section on "Case Independence" below.) Therefore, the official semantics do not "see" any bare CRLFs that are in quoted-strings; however particular parsing programs may wish to note their presence. For such programs, it would be reasonable to interpret a "CRLF LWSP-char" as being a CRLF which is part of the quoted-string; i.e., the CRLF is kept and the LWSP-char is discarded. Quoted CRLFs (i.e., a backslash followed by a CR followed by a LF) are also subject to rules of folding, but the presence of the quoting character (backslash) explicitly indicates that the CRLF is data to the quoted string. Stripping off the first following LWSP-char is also appropriate when parsing quoted CRLFs.

3.4.6. BRACKETING CHARACTERS

There is one type of bracket which must occur in matched pairs and may have pairs nested within each other:

o Parentheses ("(" and ")") are used to indicate comments.

There are three types of brackets which must occur in matched pairs, and which may NOT be nested:

o Colon/semi-colon (":" and ";") are used in address specifications to indicate that the included list of addresses are to be treated as a group.

o Angle brackets ("<" and ">") are generally used to indicate the presence of a one machine-usable reference (e.g., delimiting mailboxes), possibly including source-routing to the machine.

o Square brackets ("[" and "]") are used to indicate the presence of a domain-literal, which the appropriate name-domain is to use directly, bypassing normal name-resolution mechanisms.

3.4.7. CASE INDEPENDENCE

Except as noted, alphabetic strings may be represented in any combination of upper and lower case. The only syntactic units which requires preservation of case information are:

- text
- qtext
- dtext
- ctext
- quoted-pair
- local-part, except "Postmaster"

When matching any other syntactic unit, case is to be ignored. For example, the field-names "From", "FROM", "from", and even "FroM" are semantically equal and should all be treated identically.

When generating these units, any mix of upper and lower case alphabetic characters may be used. The case shown in this specification is suggested for message-creating processes.

Note: The reserved local-part address unit, "Postmaster", is an exception. When the value "Postmaster" is being interpreted, it must be accepted in any mixture of case, including "POSTMASTER", and "postmaster".

3.4.8. FOLDING LONG HEADER FIELDS

Each header field may be represented on exactly one line consisting of the name of the field and its body, and terminated by a CRLF; this is what the parser sees. For readability, the field-body portion of long header fields may be "folded" onto multiple lines of the actual field. "Long" is commonly interpreted to mean greater than 65 or 72 characters. The former length serves as a limit, when the message is to be viewed on most simple terminals which use simple display software; however, the limit is not imposed by this standard.

Note: Some display software often can selectively fold lines, to suit the display terminal. In such cases, sender-provided folding can interfere with the display software.

3.4.9. BACKSPACE CHARACTERS

ASCII BS characters (Backspace, decimal 8) may be included in texts and quoted-strings to effect overstriking. However, any use of backspaces which effects an overstrike to the left of the beginning of the text or quoted-string is prohibited.

3.4.10. NETWORK-SPECIFIC TRANSFORMATIONS

During transmission through heterogeneous networks, it may be necessary to force data to conform to a network's local conventions. For example, it may be required that a CR be followed either by LF, making a CRLF, or by <null>, if the CR is to stand alone). Such transformations are reversed, when the message exits that network.

When crossing network boundaries, the message should be treated as passing through two modules. It will enter the first module containing whatever network-specific transformations that were necessary to permit migration through the "current" network. It then passes through the modules:

- o Transformation Reversal

  The "current" network's idiosyncracies are removed and the message is returned to the canonical form specified in this standard.

- o Transformation

  The "next" network's local idiosyncracies are imposed on the message.

```
                     ------------------
    From     ==>    | Remove Net-A     |
    Net-A           | idiosyncracies   |
                     ------------------
                            ||
                            \/
                      Conformance
                      with standard
                            ||
                            \/
                     ------------------
                    | Impose Net-B     |  ==>    To
                    | idiosyncracies   |         Net-B
                     ------------------
```

4. MESSAGE SPECIFICATION

4.1. SYNTAX

Note: Due to an artifact of the notational conventions, the syntax indicates that, when present, some fields, must be in a particular order. Header fields are NOT required to occur in any particular order, except that the message body must occur AFTER the headers. It is recommended that, if present, headers be sent in the order "Return-Path", "Received", "Date", "From", "Subject", "Sender", "To", "cc", etc.

This specification permits multiple occurrences of most fields. Except as noted, their interpretation is not specified here, and their use is discouraged.

The following syntax for the bodies of various fields should be thought of as describing each field body as a single long string (or line). The "Lexical Analysis of Message" section on "Long Header Fields", above, indicates how such long strings can be represented on more than one line in the actual transmitted message.

```
message     =  fields *( CRLF *text )       ; Everything after
                                            ;  first null line
                                            ;  is message body

fields      =    dates                      ; Creation time,
                 source                     ;  author id & one
               1*destination                ;  address required
                *optional-field             ;  others optional

source      = [  trace ]                    ; net traversals
                 originator                 ; original mail
              [  resent ]                   ; forwarded

trace       =    return                     ; path to sender
               1*received                   ; receipt tags

return      =  "Return-path" ":" route-addr ; return address

received    =  "Received"    ":"            ; one per relay
                  ["from" domain]           ; sending host
                  ["by"   domain]           ; receiving host
                  ["via"  atom]             ; physical path
                 *("with" atom)             ; link/mail protocol
                  ["id"   msg-id]           ; receiver msg id
                  ["for"  addr-spec]        ; initial form
```

```
                   ":"    date-time          ; time received

originator   =   authentic                   ; authenticated addr
               [ "Reply-To"   ":" 1#address] )

authentic    =   "From"       ":"   mailbox  ; Single author
             / ( "Sender"     ":"   mailbox  ; Actual submittor
                 "From"       ":" 1#mailbox) ; Multiple authors
                                             ;  or not sender

resent       =   resent-authentic
               [ "Resent-Reply-To"  ":" 1#address] )

resent-authentic =
             =   "Resent-From"      ":"   mailbox
             / ( "Resent-Sender"    ":"   mailbox
                 "Resent-From"      ":" 1#mailbox  )

dates        =   orig-date                   ; Original
               [ resent-date ]               ; Forwarded

orig-date    =  "Date"        ":"   date-time

resent-date  =  "Resent-Date" ":"   date-time

destination  =  "To"          ":" 1#address  ; Primary
             /  "Resent-To"   ":" 1#address
             /  "cc"          ":" 1#address  ; Secondary
             /  "Resent-cc"   ":" 1#address
             /  "bcc"         ":"  #address  ; Blind carbon
             /  "Resent-bcc"  ":"  #address

optional-field =
             /  "Message-ID"        ":"   msg-id
             /  "Resent-Message-ID" ":"   msg-id
             /  "In-Reply-To"       ":"  *(phrase / msg-id)
             /  "References"        ":"  *(phrase / msg-id)
             /  "Keywords"          ":"  #phrase
             /  "Subject"           ":"  *text
             /  "Comments"          ":"  *text
             /  "Encrypted"         ":" 1#2word
             /  extension-field              ; To be defined
             /  user-defined-field           ; May be pre-empted

msg-id       =  "<" addr-spec ">"            ; Unique message id
```

```
extension-field =
              <Any field which is defined in a document
               published as a formal extension to this
               specification; none will have names beginning
               with the string "X-">

user-defined-field =
              <Any field which has not been defined
               in this specification or published as an
               extension to this specification; names for
               such fields must be unique and may be
               pre-empted by published extensions>
```

4.2. FORWARDING

Some systems permit mail recipients to forward a message, retaining the original headers, by adding some new fields. This standard supports such a service, through the "Resent-" prefix to field names.

Whenever the string "Resent-" begins a field name, the field has the same semantics as a field whose name does not have the prefix. However, the message is assumed to have been forwarded by an original recipient who attached the "Resent-" field. This new field is treated as being more recent than the equivalent, original field. For example, the "Resent-From", indicates the person that forwarded the message, whereas the "From" field indicates the original author.

Use of such precedence information depends upon participants' communication needs. For example, this standard does not dictate when a "Resent-From:" address should receive replies, in lieu of sending them to the "From:" address.

Note: In general, the "Resent-" fields should be treated as containing a set of information that is independent of the set of original fields. Information for one set should not automatically be taken from the other. The interpretation of multiple "Resent-" fields, of the same type, is undefined.

In the remainder of this specification, occurrence of legal "Resent-" fields are treated identically with the occurrence of fields whose names do not contain this prefix.

4.3. TRACE FIELDS

Trace information is used to provide an audit trail of message handling. In addition, it indicates a route back to the sender of the message.

The list of known "via" and "with" values are registered with the Network Information Center, SRI International, Menlo Park, California.

4.3.1. RETURN-PATH

This field is added by the final transport system that delivers the message to its recipient. The field is intended to contain definitive information about the address and route back to the message's originator.

Note: The "Reply-To" field is added by the originator and serves to direct replies, whereas the "Return-Path" field is used to identify a path back to the originator.

While the syntax indicates that a route specification is optional, every attempt should be made to provide that information in this field.

4.3.2. RECEIVED

A copy of this field is added by each transport service that relays the message. The information in the field can be quite useful for tracing transport problems.

The names of the sending and receiving hosts and time-of-receipt may be specified. The "via" parameter may be used, to indicate what physical mechanism the message was sent over, such as Arpanet or Phonenet, and the "with" parameter may be used to indicate the mail-, or connection-, level protocol that was used, such as the SMTP mail protocol, or X.25 transport protocol.

Note: Several "with" parameters may be included, to fully specify the set of protocols that were used.

Some transport services queue mail; the internal message identifier that is assigned to the message may be noted, using the "id" parameter. When the sending host uses a destination address specification that the receiving host reinterprets, by expansion or transformation, the receiving host may wish to record the original specification, using the "for" parameter. For example, when a copy of mail is sent to the member of a distribution list, this parameter may be used to record the original address that was used to specify the list.

4.4. ORIGINATOR FIELDS

The standard allows only a subset of the combinations possible with the From, Sender, Reply-To, Resent-From, Resent-Sender, and Resent-Reply-To fields. The limitation is intentional.

4.4.1. FROM / RESENT-FROM

This field contains the identity of the person(s) who wished this message to be sent. The message-creation process should default this field to be a single, authenticated machine address, indicating the AGENT (person, system or process) entering the message. If this is not done, the "Sender" field MUST be present. If the "From" field IS defaulted this way, the "Sender" field is optional and is redundant with the "From" field. In all cases, addresses in the "From" field must be machine-usable (addr-specs) and may not contain named lists (groups).

4.4.2. SENDER / RESENT-SENDER

This field contains the authenticated identity of the AGENT (person, system or process) that sends the message. It is intended for use when the sender is not the author of the message, or to indicate who among a group of authors actually sent the message. If the contents of the "Sender" field would be completely redundant with the "From" field, then the "Sender" field need not be present and its use is discouraged (though still legal). In particular, the "Sender" field MUST be present if it is NOT the same as the "From" Field.

The Sender mailbox specification includes a word sequence which must correspond to a specific agent (i.e., a human user or a computer program) rather than a standard address. This indicates the expectation that the field will identify the single AGENT (person, system, or process) responsible for sending the mail and not simply include the name of a mailbox from which the mail was sent. For example in the case of a shared login name, the name, by itself, would not be adequate. The local-part address unit, which refers to this agent, is expected to be a computer system term, and not (for example) a generalized person reference which can be used outside the network text message context.

Since the critical function served by the "Sender" field is identification of the agent responsible for sending mail and since computer programs cannot be held accountable for their behavior, it is strongly recommended that when a computer program generates a message, the HUMAN who is responsible for that program be referenced as part of the "Sender" field mailbox specification.

4.4.3. REPLY-TO / RESENT-REPLY-TO

This field provides a general mechanism for indicating any mailbox(es) to which responses are to be sent. Three typical uses for this feature can be distinguished. In the first case, the author(s) may not have regular machine-baced mailboxes and therefore wish(es) to indicate an alternate machine address. In the second case, an author may wish additional persons to be made aware of, or responsible for, replies. A somewhat different use may be of some help to "text message teleconferencing" groups equipped with automatic distribution services: include the address of that service in the "Reply-To" field of all messages submitted to the teleconference; then participants can "reply" to conference submissions to guarantee the correct distribution of any submission of their own.

Note: The "Return-Path" field is added by the mail transport service, at the time of final deliver. It is intended to identify a path back to the orginator of the message. The "Reply-To" field is added by the message originator and is intended to direct replies.

4.4.4. AUTOMATIC USE OF FROM / SENDER / REPLY-TO

For systems which automatically generate address lists for replies to messages, the following recommendations are made:

- The "Sender" field mailbox should be sent notices of any problems in transport or delivery of the original messages. If there is no "Sender" field, then the "From" field mailbox should be used.

- The "Sender" field mailbox should NEVER be used automatically, in a recipient's reply message.

- If the "Reply-To" field exists, then the reply should go to the addresses indicated in that field and not to the address(es) indicated in the "From" field.

- If there is a "From" field, but no "Reply-To" field, the reply should be sent to the address(es) indicated in the "From" field.

Sometimes, a recipient may actually wish to communicate with the person that initiated the message transfer. In such cases, it is reasonable to use the "Sender" address.

This recommendation is intended only for automated use of originator-fields and is not intended to suggest that replies may not also be sent to other recipients of messages. It is up to the respective mail-handling programs to decide what additional facilities will be provided.

Examples are provided in Appendix A.

4.5. RECEIVER FIELDS

4.5.1. TO / RESENT-TO

This field contains the identity of the primary recipients of the message.

4.5.2. CC / RESENT-CC

This field contains the identity of the secondary (informational) recipients of the message.

4.5.3. BCC / RESENT-BCC

This field contains the identity of additional recipients of the message. The contents of this field are not included in copies of the message sent to the primary and secondary recipients. Some systems may choose to include the text of the "Bcc" field only in the author(s)'s copy, while others may also include it in the text sent to all those indicated in the "Bcc" list.

4.6. REFERENCE FIELDS

4.6.1. MESSAGE-ID / RESENT-MESSAGE-ID

This field contains a unique identifier (the local-part address unit) which refers to THIS version of THIS message. The uniqueness of the message identifier is guaranteed by the host which generates it. This identifier is intended to be machine readable and not necessarily meaningful to humans. A message identifier pertains to exactly one instantiation of a particular message; subsequent revisions to the message should each receive new message identifiers.

4.6.2. IN-REPLY-TO

The contents of this field identify previous correspondence which this message answers. Note that if message identifiers are used in this field, they must use the msg-id specification format.

4.6.3. REFERENCES

The contents of this field identify other correspondence which this message references. Note that if message identifiers are used, they must use the msg-id specification format.

4.6.4. KEYWORDS

This field contains keywords or phrases, separated by commas.

4.7. OTHER FIELDS

4.7.1. SUBJECT

This is intended to provide a summary, or indicate the nature, of the message.

4.7.2. COMMENTS

Permits adding text comments onto the message without disturbing the contents of the message's body.

4.7.3. ENCRYPTED

Sometimes, data encryption is used to increase the privacy of message contents. If the body of a message has been encrypted, to keep its contents private, the "Encrypted" field can be used to note the fact and to indicate the nature of the encryption. The first <word> parameter indicates the software used to encrypt the body, and the second, optional <word> is intended to aid the recipient in selecting the proper decryption key. This code word may be viewed as an index to a table of keys held by the recipient.

Note: Unfortunately, headers must contain envelope, as well as contents, information. Consequently, it is necessary that they remain unencrypted, so that mail transport services may access them. Since names, addresses, and "Subject" field contents may contain sensitive information, this requirement limits total message privacy.

Names of encryption software are registered with the Network Information Center, SRI International, Menlo Park, California.

4.7.4. EXTENSION-FIELD

A limited number of common fields have been defined in this document. As network mail requirements dictate, additional fields may be standardized. To provide user-defined fields with a measure of safety, in name selection, such extension-fields will never have names that begin with the string "X-".

Names of Extension-fields are registered with the Network Information Center, SRI International, Menlo Park, California.

4.7.5. USER-DEFINED-FIELD

Individual users of network mail are free to define and use additional header fields. Such fields must have names which are not already used in the current specification or in any definitions of extension-fields, and the overall syntax of these user-defined-fields must conform to this specification's rules for delimiting and folding fields. Due to the extension-field publishing process, the name of a user-defined-field may be pre-empted

Note: The prefatory string "X-" will never be used in the names of Extension-fields. This provides user-defined fields with a protected set of names.

5. DATE AND TIME SPECIFICATION

5.1. SYNTAX

```
date-time   =  [ day "," ] date time        ; dd mm yy
                                            ;  hh:mm:ss zzz

day         =  "Mon"  / "Tue" /  "Wed"  / "Thu"
            /  "Fri"  / "Sat" /  "Sun"

date        =  1*2DIGIT month 2DIGIT        ; day month year
                                            ;  e.g. 20 Jun 82

month       =  "Jan"  /  "Feb" /  "Mar"  /  "Apr"
            /  "May"  /  "Jun" /  "Jul"  /  "Aug"
            /  "Sep"  /  "Oct" /  "Nov"  /  "Dec"

time        =  hour zone                    ; ANSI and Military

hour        =  2DIGIT ":" 2DIGIT [":" 2DIGIT]
                                            ; 00:00:00 - 23:59:59

zone        =  "UT"  / "GMT"                ; Universal Time
                                            ; North American : UT
            /  "EST" / "EDT"                ;  Eastern:  - 5/ - 4
            /  "CST" / "CDT"                ;  Central:  - 6/ - 5
            /  "MST" / "MDT"                ;  Mountain: - 7/ - 6
            /  "PST" / "PDT"                ;  Pacific:  - 8/ - 7
            /  1ALPHA                       ; Military: Z = UT;
                                            ;  A:-1; (J not used)
                                            ;  M:-12; N:+1; Y:+12
            / ( ("+" / "-") 4DIGIT )        ; Local differential
                                            ;  hours+min. (HHMM)
```

5.2. SEMANTICS

If included, day-of-week must be the day implied by the date specification.

Time zone may be indicated in several ways. "UT" is Universal Time (formerly called "Greenwich Mean Time"); "GMT" is permitted as a reference to Universal Time. The military standard uses a single character for each zone. "Z" is Universal Time. "A" indicates one hour earlier, and "M" indicates 12 hours earlier; "N" is one hour later, and "Y" is 12 hours later. The letter "J" is not used. The other remaining two forms are taken from ANSI standard X3.51-1975. One allows explicit indication of the amount of offset from UT; the other uses common 3-character strings for indicating time zones in North America.

6. ADDRESS SPECIFICATION

6.1. SYNTAX

```
address     =  mailbox                      ; one addressee
            /  group                        ; named list

group       =  phrase ":" [#mailbox] ";"

mailbox     =  addr-spec                    ; simple address
            /  phrase route-addr            ; name & addr-spec

route-addr  =  "<" [route] addr-spec ">"

route       =  1#("@" domain) ":"           ; path-relative

addr-spec   =  local-part "@" domain        ; global address

local-part  =  word *("." word)             ; uninterpreted
                                            ; case-preserved

domain      =  sub-domain *("." sub-domain)

sub-domain  =  domain-ref / domain-literal

domain-ref  =  atom                         ; symbolic reference
```

6.2. SEMANTICS

A mailbox receives mail. It is a conceptual entity which does not necessarily pertain to file storage. For example, some sites may choose to print mail on their line printer and deliver the output to the addressee's desk.

A mailbox specification comprises a person, system or process name reference, a domain-dependent string, and a name-domain reference. The name reference is optional and is usually used to indicate the human name of a recipient. The name-domain reference specifies a sequence of sub-domains. The domain-dependent string is uninterpreted, except by the final sub-domain; the rest of the mail service merely transmits it as a literal string.

6.2.1. DOMAINS

A name-domain is a set of registered (mail) names. A name-domain specification resolves to a subordinate name-domain specification or to a terminal domain-dependent string. Hence, domain specification is extensible, permitting any number of registration levels.

Name-domains model a global, logical, hierarchical addressing scheme. The model is logical, in that an address specification is related to name registration and is not necessarily tied to transmission path. The model's hierarchy is a directed graph, called an in-tree, such that there is a single path from the root of the tree to any node in the hierarchy. If more than one path actually exists, they are considered to be different addresses.

The root node is common to all addresses; consequently, it is not referenced. Its children constitute "top-level" name-domains. Usually, a service has access to its own full domain specification and to the names of all top-level name-domains.

The "top" of the domain addressing hierarchy -- a child of the root -- is indicated by the right-most field, in a domain specification. Its child is specified to the left, its child to the left, and so on.

Some groups provide formal registration services; these constitute name-domains that are independent logically of specific machines. In addition, networks and machines implicitly compose name-domains, since their membership usually is registered in name tables.

In the case of formal registration, an organization implements a (distributed) data base which provides an address-to-route mapping service for addresses of the form:

person@registry.organization

Note that "organization" is a logical entity, separate from any particular communication network.

A mechanism for accessing "organization" is universally available. That mechanism, in turn, seeks an instantiation of the registry; its location is not indicated in the address specification. It is assumed that the system which operates under the name "organization" knows how to find a subordinate registry. The registry will then use the "person" string to determine where to send the mail specification.

The latter, network-oriented case permits simple, direct, attachment-related address specification, such as:

user@host.network

Once the network is accessed, it is expected that a message will go directly to the host and that the host will resolve the user name, placing the message in the user's mailbox.

6.2.2. ABBREVIATED DOMAIN SPECIFICATION

Since any number of levels is possible within the domain hierarchy, specification of a fully qualified address can become inconvenient. This standard permits abbreviated domain specification, in a special case:

> For the address of the sender, call the left-most sub-domain Level N. In a header address, if all of the sub-domains above (i.e., to the right of) Level N are the same as those of the sender, then they do not have to appear in the specification. Otherwise, the address must be fully qualified.
>
> This feature is subject to approval by local sub-domains. Individual sub-domains may require their member systems, which originate mail, to provide full domain specification only. When permitted, abbreviations may be present only while the message stays within the sub-domain of the sender.
>
> Use of this mechanism requires the sender's sub-domain to reserve the names of all top-level domains, so that full specifications can be distinguished from abbreviated specifications.

For example, if a sender's address is:

```
sender@registry-A.registry-1.organization-X
```

and one recipient's address is:

```
recipient@registry-B.registry-1.organization-X
```

and another's is:

```
recipient@registry-C.registry-2.organization-X
```

then ".registry-1.organization-X" need not be specified in the the message, but "registry-C.registry-2" DOES have to be specified. That is, the first two addresses may be abbreviated, but the third address must be fully specified.

When a message crosses a domain boundary, all addresses must be specified in the full format, ending with the top-level name-domain in the right-most field. It is the responsibility of mail forwarding services to ensure that addresses conform with this requirement. In the case of abbreviated addresses, the relaying service must make the necessary expansions. It should be noted that it often is difficult for such a service to locate all occurrences of address abbreviations. For example, it will not be possible to find such abbreviations within the body of the message. The "Return-Path" field can aid recipients in recovering from these errors.

Note: When passing any portion of an addr-spec onto a process which does not interpret data according to this standard (e.g., mail protocol servers). There must be NO LWSP-chars preceding or following the at-sign or any delimiting period ("."), such as shown in the above examples, and only ONE SPACE between contiguous <word>s.

6.2.3. DOMAIN TERMS

A domain-ref must be THE official name of a registry, network, or host. It is a symbolic reference, within a name sub-domain. At times, it is necessary to bypass standard mechanisms for resolving such references, using more primitive information, such as a network host address rather than its associated host name.

To permit such references, this standard provides the domain-literal construct. Its contents must conform with the needs of the sub-domain in which it is interpreted.

Domain-literals which refer to domains within the ARPA Internet specify 32-bit Internet addresses, in four 8-bit fields noted in decimal, as described in Request for Comments #820, "Assigned Numbers." For example:

[10.0.3.19]

Note: THE USE OF DOMAIN-LITERALS IS STRONGLY DISCOURAGED. It is permitted only as a means of bypassing temporary system limitations, such as name tables which are not complete.

The names of "top-level" domains, and the names of domains under in the ARPA Internet, are registered with the Network Information Center, SRI International, Menlo Park, California.

6.2.4. DOMAIN-DEPENDENT LOCAL STRING

The local-part of an addr-spec in a mailbox specification (i.e., the host's name for the mailbox) is understood to be whatever the receiving mail protocol server allows. For example, some systems do not understand mailbox references of the form "P. D. Q. Bach", but others do.

This specification treats periods (".") as lexical separators. Hence, their presence in local-parts which are not quoted-strings, is detected. However, such occurrences carry NO semantics. That is, if a local-part has periods within it, an address parser will divide the local-part into several tokens, but the sequence of tokens will be treated as one uninterpreted unit. The sequence will be re-assembled, when the address is passed outside of the system such as to a mail protocol service.

For example, the address:

First.Last@Registry.Org

is legal and does not require the local-part to be surrounded with quotation-marks. (However, "First Last" DOES require quoting.) The local-part of the address, when passed outside of the mail system, within the Registry.Org domain, is "First.Last", again without quotation marks.

6.2.5. BALANCING LOCAL-PART AND DOMAIN

In some cases, the boundary between local-part and domain can be flexible. The local-part may be a simple string, which is used for the final determination of the recipient's mailbox. All other levels of reference are, therefore, part of the domain.

For some systems, in the case of abbreviated reference to the local and subordinate sub-domains, it may be possible to specify only one reference within the domain part and place the other, subordinate name-domain references within the local-part. This would appear as:

mailbox.sub1.sub2@this-domain

Such a specification would be acceptable to address parsers which conform to RFC #733, but do not support this newer Internet standard. While contrary to the intent of this standard, the form is legal.

Also, some sub-domains have a specification syntax which does not conform to this standard. For example:

sub-net.mailbox@sub-domain.domain

uses a different parsing sequence for local-part than for domain.

Note: As a rule, the domain specification should contain fields which are encoded according to the syntax of this standard and which contain generally-standardized information. The local-part specification should contain only that portion of the address which deviates from the form or intention of the domain field.

6.2.6. MULTIPLE MAILBOXES

An individual may have several mailboxes and wish to receive mail at whatever mailbox is convenient for the sender to access. This standard does not provide a means of specifying "any member of" a list of mailboxes.

A set of individuals may wish to receive mail as a single unit (i.e., a distribution list). The <group> construct permits specification of such a list. Recipient mailboxes are specified within the bracketed part (":" - ";"). A copy of the transmitted message is to be sent to each mailbox listed. This standard does not permit recursive specification of groups within groups.

While a list must be named, it is not required that the contents of the list be included. In this case, the <address> serves only as an indication of group distribution and would appear in the form:

name:;

Some mail services may provide a group-list distribution facility, accepting a single mailbox reference, expanding it to the full distribution list, and relaying the mail to the list's members. This standard provides no additional syntax for indicating such a service. Using the <group> address alternative, while listing one mailbox in it, can mean either that the mailbox reference will be expanded to a list or that there is a group with one member.

6.2.7. EXPLICIT PATH SPECIFICATION

At times, a message originator may wish to indicate the transmission path that a message should follow. This is called source routing. The normal addressing scheme, used in an addr-spec, is carefully separated from such information; the <route> portion of a route-addr is provided for such occasions. It specifies the sequence of hosts and/or transmission services that are to be traversed. Both domain-refs and domain-literals may be used.

Note: The use of source routing is discouraged. Unless the sender has special need of path restriction, the choice of transmission rcute should be left to the mail transport service.

6.3. RESERVED ADDRESS

It often is necessary to send mail to a site, without knowing any of its valid addresses. For example, there may be mail system dysfunctions, or a user may wish to find out a person's correct address, at that site.

This standard specifies a single, reserved mailbox address (local-part) which is to be valid at each site. Mail sent to that address is to be routed to a person responsible for the site's mail system or to a person with responsibility for general site operation. The name of the reserved local-part address is:

Postmaster

so that "Postmaster@domain" is required to be valid.

Note: This reserved local-part must be matched without sensitivity to alphabetic case, so that "POSTMASTER", "postmaster", and even "poStmASteR" is to be accepted.

7. BIBLIOGRAPHY

ANSI. "USA Standard Code for Information Interchange," X3.4. American National Standards Institute: New York (1968). Also in: Feinler, E. and J. Postel, eds., "ARPANET Protocol Handbook", NIC 7104.

ANSI. "Representations of Universal Time, Local Time Differentials, and United States Time Zone References for Information Interchange," X3.51-1975. American National Standards Institute: New York (1975).

Bemer, R.W., "Time and the Computer." In: Interface Age (Feb. 1979).

Bennett, C.J. "JNT Mail Protocol". Joint Network Team, Rutherford and Appleton Laboratory: Didcot, England.

Bhushan, A.K., Pogran, K.T., Tomlinson, R.S., and White, J.E. "Standardizing Network Mail Headers," ARPANET Request for Comments No. 561, Network Information Center No. 18516; SRI International: Menlo Park (September 1973).

Birrell, A.D., Levin, R., Needham, R.M., and Schroeder, M.D. "Grapevine: An Exercise in Distributed Computing," Communications of the ACM 25, 4 (April 1982), 260-274.

Crocker, D.H., Vittal, J.J., Pogran, K.T., Henderson, D.A. "Standard for the Format of ARPA Network Text Message," ARPANET Request for Comments No. 733, Network Information Center No. 41952. SRI International: Menlo Park (November 1977).

Feinler, E.J. and Postel, J.B. ARPANET Protocol Handbook, Network Information Center No. 7104 (NTIS AD A003890). SRI International: Menlo Park (April 1976).

Harary, F. "Graph Theory". Addison-Wesley: Reading, Mass. (1969).

Levin, R. and Schroeder, M. "Transport of Electronic Messages through a Network," TeleInformatics 79, pp. 29-33. North Holland (1979). Also as Xerox Palo Alto Research Center Technical Report CSL-79-4.

Myer, T.H. and Henderson, D.A. "Message Transmission Protocol," ARPANET Request for Comments, No. 680, Network Information Center No. 32116. SRI International: Menlo Park (1975).

NBS. "Specification of Message Format for Computer Based Message Systems, Recommended Federal Information Processing Standard." National Bureau of Standards: Gaithersburg, Maryland (October 1981).

NIC. Internet Protocol Transition Workbook. Network Information Center, SRI-International, Menlo Park, California (March 1982).

Oppen, D.C. and Dalal, Y.K. "The Clearinghouse: A Decentralized Agent for Locating Named Objects in a Distributed Environment," OPD-T8103. Xerox Office Products Division: Palo Alto, CA. (October 1981).

Postel, J.B. "Assigned Numbers," ARPANET Request for Comments, No. 820. SRI International: Menlo Park (August 1982).

Postel, J.B. "Simple Mail Transfer Protocol," ARPANET Request for Comments, No. 821. SRI International: Menlo Park (August 1982).

Shoch, J.F. "Internetwork naming, addressing and routing," in Proc. 17th IEEE Computer Society International Conference, pp. 72-79, Sept. 1978, IEEE Cat. No. 78 CH 1388-8C.

Su, Z. and Postel, J. "The Domain Naming Convention for Internet User Applications," ARPANET Request for Comments, No. 819. SRI International: Menlo Park (August 1982).

APPENDIX

A. EXAMPLES

A.1. ADDRESSES

A.1.1. Alfred Neuman <Neuman@BBN-TENEXA>

A.1.2. Neuman@BBN-TENEXA

These two "Alfred Neuman" examples have identical semantics, as far as the operation of the local host's mail sending (distribution) program (also sometimes called its "mailer") and the remote host's mail protocol server are concerned. In the first example, the "Alfred Neuman" is ignored by the mailer, as "Neuman@BBN-TENEXA" completely specifies the recipient. The second example contains no superfluous information, and, again, "Neuman@BBN-TENEXA" is the intended recipient.

Note: When the message crosses name-domain boundaries, then these specifications must be changed, so as to indicate the remainder of the hierarchy, starting with the top level.

A.1.3. "George, Ted" <Shared@Group.Arpanet>

This form might be used to indicate that a single mailbox is shared by several users. The quoted string is ignored by the originating host's mailer, because "Shared@Group.Arpanet" completely specifies the destination mailbox.

A.1.4. Wilt . (the Stilt) Chamberlain@NBA.US

The "(the Stilt)" is a comment, which is NOT included in the destination mailbox address handed to the originating system's mailer. The local-part of the address is the string "Wilt.Chamberlain", with NO space between the first and second words.

A.1.5. Address Lists

```
Gourmets:  Pompous Person <WhoZiWhatZit@Cordon-Bleu>,
           Childs@WGBH.Boston, Galloping Gourmet@
           ANT.Down-Under (Australian National Television),
           Cheapie@Discount-Liquors;,
  Cruisers:  Port@Portugal, Jones@SEA;,
    Another@Somewhere.SomeOrg
```

This group list example points out the use of comments and the mixing of addresses and groups.

A.2. ORIGINATOR ITEMS

A.2.1. Author-sent

George Jones logs into his host as "Jones". He sends mail himself.

```
From:  Jones@Group.Org
```

or

```
From:  George Jones <Jones@Group.Org>
```

A.2.2. Secretary-sent

George Jones logs in as Jones on his host. His secretary, who logs in as Secy sends mail for him. Replies to the mail should go to George.

```
From:    George Jones <Jones@Group>
Sender:  Secy@Other-Group
```

A.2.3. Secretary-sent, for user of shared directory

George Jones' secretary sends mail for George. Replies should go to George.

```
From:     George Jones<Shared@Group.Org>
Sender:   Secy@Other-Group
```

Note that there need not be a space between "Jones" and the "<", but adding a space enhances readability (as is the case in other examples.

A.2.4. Committee activity, with one author

George is a member of a committee. He wishes to have any replies to his message go to all committee members.

```
From:     George Jones <Jones@Host.Net>
Sender:   Jones@Host
Reply-To: The Committee: Jones@Host.Net,
                         Smith@Other.Org,
                         Doe@Somewhere-Else;
```

Note that if George had not included himself in the enumeration of The Committee, he would not have gotten an implicit reply; the presence of the "Reply-to" field SUPERSEDES the sending of a reply to the person named in the "From" field.

A.2.5. Secretary acting as full agent of author

George Jones asks his secretary (Secy@Host) to send a message for him in his capacity as Group. He wants his secretary to handle all replies.

```
From:     George Jones <Group@Host>
Sender:   Secy@Host
Reply-To: Secy@Host
```

A.2.6. Agent for user without online mailbox

A friend of George's, Sarah, is visiting. George's secretary sends some mail to a friend of Sarah in computerland. Replies should go to George, whose mailbox is Jones at Registry.

```
From:     Sarah Friendly <Secy@Registry>
Sender:   Secy-Name <Secy@Registry>
Reply-To: Jones@Registry.
```

A.2.7. Agent for member of a committee

George's secretary sends out a message which was authored jointly by all the members of a committee. Note that the name of the committee cannot be specified, since <group> names are not permitted in the From field.

```
From:   Jones@Host,
        Smith@Other-Host,
        Doe@Somewhere-Else
Sender: Secy@SHost
```

A.3. COMPLETE HEADERS

A.3.1. Minimum required

```
Date:     26 Aug 76 1429 EDT        Date:     26 Aug 76 1429 EDT
From:     Jones@Registry.Org   or   From:     Jones@Registry.Org
Bcc:                                To:       Smith@Registry.Org
```

Note that the "Bcc" field may be empty, while the "To" field is required to have at least one address.

A.3.2. Using some of the additional fields

```
Date:     26 Aug 76 1430 EDT
From:     George Jones<Group@Host>
Sender:   Secy@SHOST
To:       "Al Neuman"@Mad-Host,
          Sam.Irving@Other-Host
Message-ID:  <some.string@SHOST>
```

A.3.3. About as complex as you're going to get

```
Date     :  27 Aug 76 0932 PDT
From     :  Ken Davis <KDavis@This-Host.This-net>
Subject  :  Re: The Syntax in the RFC
Sender   :  KSecy@Other-Host
Reply-To :  Sam.Irving@Reg.Organization
To       :  George Jones <Group@Some-Reg.An-Org>,
            Al.Neuman@MAD.Publisher
cc       :  Important folk:
              Tom Softwood <Balsa@Tree.Root>,
              "Sam Irving"@Other-Host;,
            Standard Distribution:
              /main/davis/people/standard@Other-Host,
              "<Jones>standard.dist.3"@Tops-20-Host>;
Comment  :  Sam is away on business. He asked me to handle
            his mail for him.  He'll be able to provide  a
            more  accurate  explanation  when  he  returns
            next week.
In-Reply-To: <some.string@DBM.Group>, George's message
X-Special-action:  This is a sample of user-defined field-
            names.  There could also be a field-name
            "Special-action", but its name might later be
            preempted
Message-ID: <4231.629.XYzi-What@Other-Host>
```

B. SIMPLE FIELD PARSING

Some mail-reading software systems may wish to perform only minimal processing, ignoring the internal syntax of structured field-bodies and treating them the same as unstructured-field-bodies. Such software will need only to distinguish:

- o Header fields from the message body,
- o Beginnings of fields from lines which continue fields,
- o Field-names from field-contents.

[illegible] abbreviated set of syntactic rules which follows will suffice for this purpose. It describes a limited view of messages and is a subset of the syntactic rules provided in the main part of this specification. One small exception is that the contents of field-bodies consist only of text:

B.1. SYNTAX

```
message         =  *field *(CRLF *text)

field           =   field-name ":" [field-body] CRLF

field-name      =  1*<any CHAR, excluding CTLs, SPACE, and ":">

field-body      =   *text [CRLF LWSP-char field-body]
```

B.2. SEMANTICS

Headers occur before the message body and are terminated by a null line (i.e., two contiguous CRLFs).

A line which continues a header field begins with a SPACE or HTAB character, while a line beginning a field starts with a printable character which is not a colon.

A field-name consists of one or more printable characters (excluding colon, space, and control-characters). A field-name MUST be contained on one line. Upper and lower case are not distinguished when comparing field-names.

C. DIFFERENCES FROM RFC #733

The following summarizes the differences between this standard and the one specified in Arpanet Request for Comments #733, "Standard for the Format of ARPA Network Text Messages". The differences are listed in the order of their occurrence in the current specification.

C.1. FIELD DEFINITIONS

C.1.1. FIELD NAMES

These now must be a sequence of printable characters. They may not contain any LWSP-chars.

C.2. LEXICAL TOKENS

C.2.1. SPECIALS

The characters period ("."), left-square bracket ("["), and right-square bracket ("]") have been added. For presentation purposes, and when passing a specification to a system that does not conform to this standard, periods are to be contiguous with their surrounding lexical tokens. No linear-whitespace is permitted between them. The presence of one LWSP-char between other tokens is still directed.

C.2.2. ATOM

Atoms may not contain SPACE.

C.2.3. SPECIAL TEXT

ctext and qtext have had backslash ("\") added to the list of prohibited characters.

C.2.4. DOMAINS

The lexical tokens <domain-literal> and <dtext> have been added.

C.3. MESSAGE SPECIFICATION

C.3.1. TRACE

The "Return-path:" and "Received:" fields have been specified.

C.3.2. FROM

The "From" field must contain machine-usable addresses (addr-spec). Multiple addresses may be specified, but named-lists (groups) may not.

C.3.3. RESENT

The meta-construct of prefacing field names with the string "Resent-" has been added, to indicate that a message has been forwarded by an intermediate recipient.

C.3.4. DESTINATION

A message must contain at least one destination address field. "To" and "CC" are required to contain at least one address.

C.3.5. IN-REPLY-TO

The field-body is no longer a comma-separated list, although a sequence is still permitted.

C.3.6. REFERENCE

The field-body is no longer a comma-separated list, although a sequence is still permitted.

C.3.7. ENCRYPTED

A field has been specified that permits senders to indicate that the body of a message has been encrypted.

C.3.8. EXTENSION-FIELD

Extension fields are prohibited from beginning with the characters "X-".

C.4. DATE AND TIME SPECIFICATION

C.4.1. SIMPLIFICATION

Fewer optional forms are permitted and the list of three-letter time zones has been shortened.

C.5. ADDRESS SPECIFICATION

C.5.1. ADDRESS

The use of quoted-string, and the ":"-atom-":" construct, have been removed. An address now is either a single mailbox reference or is a named list of addresses. The latter indicates a group distribution.

C.5.2. GROUPS

Group lists are now required to to have a name. Group lists may not be nested.

C.5.3. MAILBOX

A mailbox specification may indicate a person's name, as before. Such a named list no longer may specify multiple mailboxes and may not be nested.

C.5.4. ROUTE ADDRESSING

Addresses now are taken to be absolute, global specifications, independent of transmission paths. The <route> construct has been provided, to permit explicit specification of transmission path. RFC #733's use of multiple at-signs ("@") was intended as a general syntax for indicating routing and/or hierarchical addressing. The current standard separates these specifications and only one at-sign is permitted.

C.5.5. AT-SIGN

The string " at " no longer is used as an address delimiter. Only at-sign ("@") serves the function.

C.5.6. DOMAINS

Hierarchical, logical name-domains have been added.

C.6. RESERVED ADDRESS

The local-part "Postmaster" has been reserved, so that users can be guaranteed at least one valid address at a site.

D. ALPHABETICAL LISTING OF SYNTAX RULES

```
address     =  mailbox                      ; one addressee
            /  group                        ; named list
addr-spec   =  local-part "@" domain        ; global address
ALPHA       =  <any ASCII alphabetic character>
                                            ; (101-132, 65.- 90.)
                                            ; (141-172, 97.-122.)
atom        =  1*<any CHAR except specials, SPACE and CTLs>
authentic   =   "From"       ":"   mailbox  ; Single author
            / ( "Sender"     ":"   mailbox  ; Actual submittor
                "From"       ":" 1#mailbox) ; Multiple authors
                                            ;  or not sender
CHAR        =  <any ASCII character>        ; (  0-177,  0.-127.)
comment     =  "(" *(ctext / quoted-pair / comment) ")"
CR          =  <ASCII CR, carriage return>  ; (     15,      13.)
CRLF        =  CR LF
ctext       =  <any CHAR excluding "(",     ; => may be folded
                ")", "\" & CR, & including
                linear-white-space>
CTL         =  <any ASCII control           ; (  0- 37,  0.- 31.)
                character and DEL>          ; (    177,     127.)
date        =  1*2DIGIT month 2DIGIT        ; day month year
                                            ;  e.g. 20 Jun 82
dates       =   orig-date                   ; Original
              [ resent-date ]               ; Forwarded
date-time   =  [ day "," ] date time        ; dd mm yy
                                            ;  hh:mm:ss zzz
day         =  "Mon"  / "Tue" /  "Wed"  / "Thu"
            /  "Fri"  / "Sat" /  "Sun"
delimiters  =  specials / linear-white-space / comment
destination =  "To"          ":" 1#address  ; Primary
            /  "Resent-To"   ":" 1#address
            /  "cc"          ":" 1#address  ; Secondary
            /  "Resent-cc"   ":" 1#address
            /  "bcc"         ":"  #address  ; Blind carbon
            /  "Resent-bcc"  ":"  #address
DIGIT       =  <any ASCII decimal digit>    ; ( 60- 71, 48.- 57.)
domain      =  sub-domain *("." sub-domain)
domain-literal =  "[" *(dtext / quoted-pair) "]"
domain-ref  =  atom                         ; symbolic reference
dtext       =  <any CHAR excluding "[",     ; => may be folded
                "]", "\" & CR, & including
                linear-white-space>
extension-field =
              <Any field which is defined in a document
               published as a formal extension to this
               specification; none will have names beginning
               with the string "X-">
```

```
field       =  field-name ":" [ field-body ] CRLF
fields      =    dates                          ; Creation time,
                 source                         ;  author id & one
               1*destination                    ;  address required
                *optional-field                 ;  others optional
field-body  =  field-body-contents
               [CRLF LWSP-char field-body]
field-body-contents =
              <the ASCII characters making up the field-body, as
               defined in the following sections, and consisting
               of combinations of atom, quoted-string, and
               specials tokens, or else consisting of texts>
field-name  =  1*<any CHAR, excluding CTLs, SPACE, and ":">
group       =  phrase ":" [#mailbox] ";"
hour        =  2DIGIT ":" 2DIGIT [":" 2DIGIT]
                                                ; 00:00:00 - 23:59:59
HTAB        =  <ASCII HT, horizontal-tab>       ; (      11,       9.)
LF          =  <ASCII LF, linefeed>             ; (      12,      10.)
linear-white-space =  1*([CRLF] LWSP-char)      ; semantics = SPACE
                                                ; CRLF => folding
local-part  =  word *("." word)                 ; uninterpreted
                                                ; case-preserved
LWSP-char   =  SPACE / HTAB                     ; semantics = SPACE
mailbox     =  addr-spec                        ; simple address
            /  phrase route-addr                ; name & addr-spec
message     =  fields *( CRLF *text )           ; Everything after
                                                ;  first null line
                                                ;  is message body
month       =  "Jan"  /  "Feb" /  "Mar"  /  "Apr"
            /  "May"  /  "Jun" /  "Jul"  /  "Aug"
            /  "Sep"  /  "Oct" /  "Nov"  /  "Dec"
msg-id      =  "<" addr-spec ">"                ; Unique message id
optional-field =
            /  "Message-ID"        ":"   msg-id
            /  "Resent-Message-ID" ":"   msg-id
            /  "In-Reply-To"       ":"  *(phrase / msg-id)
            /  "References"        ":"  *(phrase / msg-id)
            /  "Keywords"          ":"  #phrase
            /  "Subject"           ":"  *text
            /  "Comments"          ":"  *text
            /  "Encrypted"         ":" 1#2word
            /  extension-field              ; To be defined
            /  user-defined-field           ; May be pre-empted
orig-date   =  "Date"        ":"   date-time
originator  =   authentic                       ; authenticated addr
              [ "Reply-To"   ":" 1#address] )
phrase      =  1*word                           ; Sequence of words
```

```
qtext       =  <any CHAR excepting <">,     ; => may be folded
                "\" & CR, and including
                linear-white-space>
quoted-pair =  "\" CHAR                     ; may quote any char
quoted-string = <"> *(qtext/quoted-pair) <">; Regular qtext or
                                            ;   quoted chars.
received    =  "Received"    ":"            ; one per relay
                  ["from" domain]           ; sending host
                  ["by"   domain]           ; receiving host
                  ["via"  atom]             ; physical path
                 *("with" atom)             ; link/mail protocol
                  ["id"   msg-id]           ; receiver msg id
                  ["for"  addr-spec]        ; initial form
                   ";"    date-time         ; time received

resent      =   resent-authentic
              [ "Resent-Reply-To"  ":" 1#address] )
resent-authentic =
            =   "Resent-From"      ":"   mailbox
            / ( "Resent-Sender"    ":"   mailbox
                "Resent-From"      ":" 1#mailbox  )
resent-date =  "Resent-Date" ":"   date-time
return      =  "Return-path" ":" route-addr ; return address
route       =  1#("@" domain) ":"           ; path-relative
route-addr  =  "<" [route] addr-spec ">"
source      = [  trace ]                    ; net traversals
                 originator                 ; original mail
              [  resent ]                   ; forwarded
SPACE       =  <ASCII SP, space>            ; (     40,      32.)
specials    =  "(" / ")" / "<" / ">" / "@"  ; Must be in quoted-
            /  "," / ";" / ":" / "\" / <">  ;  string, to use
            /  "." / "[" / "]"              ;  within a word.
sub-domain  =  domain-ref / domain-literal
text        =  <any CHAR, including bare    ; => atoms, specials,
                CR & bare LF, but NOT       ;  comments and
                including CRLF>             ;  quoted-strings are
                                            ;  NOT recognized.
time        =  hour zone                    ; ANSI and Military
trace       =    return                     ; path to sender
               1*received                   ; receipt tags
user-defined-field =
              <Any field which has not been defined
               in this specification or published as an
               extension to this specification; names for
               such fields must be unique and may be
               pre-empted by published extensions>
word        =  atom / quoted-string
```

```
zone        =  "UT"  / "GMT"                ; Universal Time
                                            ; North American : UT
            /  "EST" / "EDT"                ;  Eastern:  - 5/ - 4
            /  "CST" / "CDT"                ;  Central:  - 6/ - 5
            /  "MST" / "MDT"                ;  Mountain: - 7/ - 6
            /  "PST" / "PDT"                ;  Pacific:  - 8/ - 7
            /  1ALPHA                       ; Military: Z = UT;
<">         =  <ASCII quote mark>           ; (     42,      34.)
```